AWS CloudShell

User Guide
AWS CloudShell: User Guide
Copyright © 2023 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.
# Table of Contents

What is AWS CloudShell? ............................................................................................................. 1
AWS CloudShell features .................................................................................................................. 1
  AWS Command Line Interface ........................................................................................................ 1
  Shells and development tools ....................................................................................................... 2
  Persistent storage ........................................................................................................................... 2
  Security .......................................................................................................................................... 2
  Customization options .................................................................................................................... 2
  Session restore ............................................................................................................................... 3
Pricing for AWS CloudShell ............................................................................................................. 3
How to get started with AWS CloudShell? ...................................................................................... 3
Key AWS CloudShell topics ........................................................................................................... 5
FAQs .................................................................................................................................................. 6
  How do I get started using AWS CloudShell? ............................................................................... 6
  What do I need to access AWS CloudShell? .................................................................................. 6
  What is AWS CloudShell on the Console Toolbar? ....................................................................... 7
  How do I launch AWS CloudShell on the Console Toolbar? .......................................................... 7
  Which AWS Regions is AWS CloudShell available in? .................................................................. 7
  Which AWS Region is assigned if AWS CloudShell isn't available in the selected Region when you launch CloudShell on the Console Toolbar? ................................................................. 7
  What types of shells can I use in AWS CloudShell? ..................................................................... 7
  What web browsers can I use with AWS CloudShell? ................................................................. 7
  Which web browsers can I use when I launch AWS CloudShell on the Console Toolbar? .......... 8
  Can I download a file when I launch AWS CloudShell on the Console Toolbar? ......................... 8
  What software is pre-installed on my shell environment? ............................................................ 8
  Can I install software that's not available in the shell environment? .......................................... 8
  Can I restrict the actions that users can do in AWS CloudShell? .................................................. 8
  How can I move data from my home directory if I want to change the AWS Region where I'm using AWS CloudShell? .............................................................................................................. 9
  Can I increase the limit that determines when AWS CloudShell times out because of user inactivity? .............................................................................................................................................. 9
  Can I access AWS CloudShell in the AWS Console Mobile Application from the home screen? ........................................................................................................................................... 9
  How can I launch AWS CloudShell in the AWS Console Mobile Application? ........................... 9
  Can I use modifier keys on my iOS and Android keyboards when using AWS CloudShell in the AWS Console Mobile Application? ..................................................................................... 10
  Can I split the AWS CloudShell tab display into multiple tabs on the AWS Console Mobile Application? ........................................................................................................................................... 10
  Can I access AWS CloudShell on the Console Toolbar on a mobile device? ............................... 10
Getting started .................................................................................................................................. 11
Prerequisites ................................................................................................................................... 11
Contents .......................................................................................................................................... 11
Step 1: Sign in to AWS Management Console .................................................................................. 12
Step 2: Select a Region, launch AWS CloudShell, and choose a shell ........................................... 14
Step 3: Download a file from AWS CloudShell .............................................................................. 15
Step 4: Upload a file to AWS CloudShell ....................................................................................... 16
Step 5: Remove a file from AWS CloudShell .................................................................................. 17
Step 6: Create a home directory backup .......................................................................................... 17
Step 7: Restart a shell session .......................................................................................................... 18
Step 8: Delete a shell session home directory ................................................................................... 19
Step 9: Edit your file's code and run it using the command line ..................................................... 20
Step 10: Use AWS CLI to add the file as an object in an Amazon S3 bucket ................................. 21
Related topics ................................................................................................................................... 22
Tutorials .......................................................................................................................................... 23
  Tutorial: Copying multiple files ...................................................................................................... 23
  Uploading and downloading multiple files using Amazon S3 ....................................................... 23
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uploading and downloading multiple files using zipped folders</td>
<td>25</td>
</tr>
<tr>
<td>Tutorial: Using CodeCommit</td>
<td>26</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>26</td>
</tr>
<tr>
<td>Step 1: Create and clone a CodeCommit repository</td>
<td>27</td>
</tr>
<tr>
<td>Step 2: Stage and commit a file before pushing it to your CodeCommit repository</td>
<td>27</td>
</tr>
<tr>
<td>Tutorial: Creating presigned URLs</td>
<td>28</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>28</td>
</tr>
<tr>
<td>Step 1: Create an IAM role to grant access to Amazon S3 bucket</td>
<td>29</td>
</tr>
<tr>
<td>Generate the presigned URL</td>
<td>29</td>
</tr>
<tr>
<td>Working with AWS CloudShell</td>
<td>31</td>
</tr>
<tr>
<td>Navigating the AWS CloudShell interface</td>
<td>31</td>
</tr>
<tr>
<td>Working in AWS Regions</td>
<td>32</td>
</tr>
<tr>
<td>Specifying your default AWS Region for AWS CLI</td>
<td>32</td>
</tr>
<tr>
<td>Working with files and storage</td>
<td>33</td>
</tr>
<tr>
<td>Accessibility features</td>
<td>34</td>
</tr>
<tr>
<td>Keyboard navigation in CloudShell</td>
<td>34</td>
</tr>
<tr>
<td>CloudShell terminal accessibility features</td>
<td>34</td>
</tr>
<tr>
<td>Choosing font sizes and interface themes in CloudShell</td>
<td>34</td>
</tr>
<tr>
<td>Working with AWS services</td>
<td>35</td>
</tr>
<tr>
<td>AWS CLI command line examples for selected AWS services</td>
<td>35</td>
</tr>
<tr>
<td>DynamoDB</td>
<td>35</td>
</tr>
<tr>
<td>AWS Cloud9</td>
<td>36</td>
</tr>
<tr>
<td>Amazon EC2</td>
<td>36</td>
</tr>
<tr>
<td>S3 Glacier</td>
<td>36</td>
</tr>
<tr>
<td>AWS Elastic Beanstalk CLI</td>
<td>36</td>
</tr>
<tr>
<td>Amazon ECS CLI</td>
<td>37</td>
</tr>
<tr>
<td>AWS SAM CLI</td>
<td>37</td>
</tr>
<tr>
<td>Customizing AWS CloudShell</td>
<td>38</td>
</tr>
<tr>
<td>Splitting the command line display into multiple tabs</td>
<td>38</td>
</tr>
<tr>
<td>Changing font size</td>
<td>38</td>
</tr>
<tr>
<td>Changing the interface theme</td>
<td>39</td>
</tr>
<tr>
<td>Using Safe Paste for multilime text</td>
<td>39</td>
</tr>
<tr>
<td>Using tmux to sesson restore</td>
<td>39</td>
</tr>
<tr>
<td>Security</td>
<td>40</td>
</tr>
<tr>
<td>Data protection</td>
<td>40</td>
</tr>
<tr>
<td>Data encryption</td>
<td>41</td>
</tr>
<tr>
<td>Identity and access management</td>
<td>41</td>
</tr>
<tr>
<td>Audience</td>
<td>42</td>
</tr>
<tr>
<td>Authentication</td>
<td>42</td>
</tr>
<tr>
<td>Managing AWS CloudShell access and usage with IAM policies</td>
<td>43</td>
</tr>
<tr>
<td>Logging and monitoring</td>
<td>47</td>
</tr>
<tr>
<td>Monitoring activity with CloudTrail</td>
<td>47</td>
</tr>
<tr>
<td>AWS CloudShell in CloudTrail</td>
<td>47</td>
</tr>
<tr>
<td>Compliance validation</td>
<td>48</td>
</tr>
<tr>
<td>Resilience</td>
<td>51</td>
</tr>
<tr>
<td>Infrastructure security</td>
<td>52</td>
</tr>
<tr>
<td>Configuration and vulnerability analysis</td>
<td>52</td>
</tr>
<tr>
<td>Security best practices</td>
<td>52</td>
</tr>
<tr>
<td>Security FAQs</td>
<td>53</td>
</tr>
<tr>
<td>What are the AWS processes and technologies that are used when you launch CloudShell and start a shell session?</td>
<td>53</td>
</tr>
<tr>
<td>Is it possible to restrict network access to CloudShell?</td>
<td>53</td>
</tr>
<tr>
<td>Can I customize my CloudShell environment?</td>
<td>53</td>
</tr>
<tr>
<td>Where is my $HOME directory actually stored in the AWS Cloud?</td>
<td>54</td>
</tr>
<tr>
<td>Is it possible to encrypt my $HOME directory?</td>
<td>54</td>
</tr>
<tr>
<td>Can I run a virus scan on my $HOME directory?</td>
<td>54</td>
</tr>
</tbody>
</table>
AWS CloudShell compute environment ................................................................. 55
  Compute environment resources ........................................................................ 55
  CloudShell network requirements ...................................................................... 55
  Pre-installed software ....................................................................................... 56
    Shells ............................................................................................................... 56
    AWS command line interfaces (CLI) ............................................................... 56
    Runtimes and AWS SDKs: Node.js and Python 3 ........................................ 58
    Development tools and shell utilities .......................................................... 59
  Installing AWS CLI to your home directory ..................................................... 63
  Installing third-party software on your shell environment ............................. 64
  Modifying your shell with scripts ..................................................................... 64
  Migrating from Amazon Linux 2 to Amazon Linux 2023 ................................. 65
  AWS CloudShell Migration FAQs .................................................................... 65
Troubleshooting ................................................................................................. 67
  Troubleshooting errors .................................................................................... 67
    Unable to start the environment. To retry, refresh the browser or restart by selecting Actions, Restart AWS CloudShell ......................................................... 67
    Unable to start the environment. You don't have required permissions. Ask your IAM administrator to grant access to AWS CloudShell ........................................ 67
    Unable to access AWS CloudShell command line ........................................ 68
    Unable to ping external IP addresses ............................................................. 68
    There were some issues preparing your terminal ......................................... 68
    Arrow keys not working correctly in PowerShell .......................................... 68
    Unsupported Web Sockets cause a failure to start CloudShell sessions ......... 69
    Unable to import the AWSPowerShell.NetCore module .............................. 70
Supported browsers .......................................................................................... 71
Supported Regions ............................................................................................. 72
  GovCloud Regions ........................................................................................... 72
  Opt-in Regions ................................................................................................. 72
Service quotas and restrictions .......................................................................... 74
  Persistent storage ............................................................................................. 74
  Monthly usage ................................................................................................. 74
  Command size ................................................................................................. 75
  Concurrent shells ............................................................................................ 75
Shell sessions ..................................................................................................... 75
Network access and data transfer ....................................................................... 75
Restrictions on system files and page reloads .................................................... 75
Document history ............................................................................................... 77
What is AWS CloudShell?

AWS CloudShell is a browser-based, pre-authenticated shell that you can launch directly from the AWS Management Console. You can navigate to CloudShell from the AWS Management Console a few different ways. For more information, see How to get started with AWS CloudShell? (p. 3)

You can run AWS CLI commands using your preferred shell, such as Bash, PowerShell, or Z shell. And you can do this without downloading or installing command line tools.

When you launch AWS CloudShell, a compute environment (p. 55) that's based on Amazon Linux 2 is created. Within this environment, you can access an extensive range of pre-installed development tools (p. 56), options for uploading (p. 16) and downloading (p. 15) files, and file storage that persists between sessions (p. 2).

(Try it now: Getting started with AWS CloudShell (p. 11))

AWS CloudShell features

This topic describes how to launch CloudShell from the Console, seamlessly switch between your preferred command line shells, and customize CloudShell to your exact preference. Additionally, you can use up to 1 GB of persistent storage in each AWS Region, and how CloudShell environment is protected by specific security features.

AWS Command Line Interface

You can launch AWS CloudShell from the AWS Management Console. The AWS credentials that you used to sign in to the console are automatically available in a new shell session. Because AWS CloudShell users are pre-authenticated, you don't need to configure credentials when interacting with AWS services using AWS CLI version 2. The AWS CLI is pre-installed on the shell's compute environment.
For more information about interacting with AWS services using the command line interface, see Working with AWS services in AWS CloudShell (p. 35).

**Shells and development tools**

With the shell that's created for AWS CloudShell sessions, you can switch seamlessly between your preferred command line shells. More specifically, you can switch between Bash, PowerShell, and Z shell. You also have access to pre-installed tools and utilities. These include git, make, pip, sudo, tar, tmux, vim, wget, and zip.

The shell environment is pre-configured with support for several leading major software languages, such as Node.js and Python. This means that, for example, you can run Node.js and Python projects without first performing runtime installations. PowerShell users can use the .NET Core runtime.

You can commit files that are created in or uploaded to AWS CloudShell to a local repository before pushing these files to a remote repository that's managed by AWS CodeCommit.

For more information, see AWS CloudShell compute environment: specifications and software (p. 55).

**Persistent storage**

With AWS CloudShell, you can use up to 1 GB of persistent storage in each AWS Region at no additional cost. Persistent storage is located in your home directory ($HOME) and is private to you. Unlike ephemeral environment resources that are recycled after each shell session ends, data in your home directory persists between sessions.

For more information about the retention of data in persistent storage, see Persistent storage (p. 74).

**Security**

The AWS CloudShell environment and its users are protected by specific security features. This includes such features as IAM permissions management, shell session restrictions, and Safe Paste for text input.

**Permissions management with IAM**

As administrator, you can grant and deny permissions to AWS CloudShell users using IAM policies. You can also create policies that specify the particular actions that users can perform with the shell environment. For more information, see Managing AWS CloudShell access and usage with IAM policies (p. 43).

**Shell session management**

Inactive and long-running sessions are automatically stopped and recycled. For more information, see Shell sessions (p. 75).

**Safe Paste for text input**

Safe Paste is enabled by default. This security feature requires that you verify that the multiline text that you want to paste into the shell doesn't contain malicious scripts. For more information, see Using Safe Paste for multiline text (p. 39).

**Customization options**

You can customize your AWS CloudShell experience to your exact preference. For example, you can change the screen layouts (multiple tabs), displayed text sizes, and toggle between the light and dark interface themes. For more information, see Customizing your AWS CloudShell experience (p. 38).
You can also extend your shell environment by installing your own software (p. 64) and modifying start-up shell scripts (p. 64).

Session restore

The session restore functionality restores sessions that you were running across single or multiple browser tabs in the CloudShell terminal. If you refresh or reopen recently closed browser tabs, this functionality resumes the session until the shell is stopped because of inactive session. To continue using your CloudShell session, press any key within the terminal window. For more information about Shell sessions, see Shell sessions (p. 75).

Session restore also restores the latest terminal output and running processes in each terminal tabs.

Note
Session restore isn't available in mobile applications.

Pricing for AWS CloudShell

AWS CloudShell is an AWS service that's available at no additional charge. However, you pay for other AWS resources that you run with AWS CloudShell. Moreover, standard data transfer rates also apply. For more information, see AWS CloudShell pricing.

For more information, see Service quotas and restrictions for AWS CloudShell (p. 74).

How to get started with AWS CloudShell?

To start working with the shell, sign in to the AWS Management Console and choose one of the following options:

- On the navigation bar, choose the CloudShell icon.
- In the Search box, type “CloudShell”, and then choose CloudShell.

This step opens your CloudShell session to a full screen.
In the Recently visited widget, choose CloudShell. This step opens your CloudShell session to a full screen.

Choose CloudShell on the Console Toolbar, on the lower left of the console. You can adjust the height of your CloudShell session by dragging =.
You can also switch your CloudShell session to a full screen by clicking **Open in new browser tab**.

For instructions on how to sign in to the AWS Management Console and performing key tasks with AWS CloudShell, see **Getting started with AWS CloudShell (p. 11)**.

**Key AWS CloudShell topics**

- Getting started with AWS CloudShell (p. 11)
- Working with AWS CloudShell (p. 31)
- Working with AWS services in AWS CloudShell (p. 35)
- Customizing your AWS CloudShell experience (p. 38)
- AWS CloudShell compute environment: specifications and software (p. 55)
AWS CloudShell FAQs

The following are answers to some common questions about CloudShell.

For more FAQs focused on security, see AWS CloudShell Security FAQs (p. 53).

- How do I get started using AWS CloudShell? (p. 6)
- What do I need to access AWS CloudShell? (p. 6)
- What is AWS CloudShell on the Console Toolbar? (p. 7)
- How do I launch AWS CloudShell on the Console Toolbar? (p. 7)
- Which AWS Regions is AWS CloudShell available in? (p. 7)
- Which AWS Region is assigned if AWS CloudShell isn't available in the selected Region when you launch CloudShell on the Console Toolbar? (p. 7)
- What types of shells can I use in AWS CloudShell? (p. 7)
- What web browsers can I use with AWS CloudShell? (p. 7)
- Which web browsers can I use when I launch AWS CloudShell on the Console Toolbar? (p. 8)
- Can I download a file when I launch AWS CloudShell on the Console Toolbar? (p. 8)
- What software is pre-installed on my shell environment? (p. 8)
- Can I install software that's not available in the shell environment? (p. 8)
- Can I restrict the actions that users can do in AWS CloudShell? (p. 8)
- How can I move data from my home directory if I want to change the AWS Region where I'm using AWS CloudShell? (p. 9)
- Can I increase the limit that determines when AWS CloudShell times out because of user inactivity? (p. 9)
- Can I access AWS CloudShell in the AWS Console Mobile Application from the home screen? (p. 9)
- How can I launch AWS CloudShell in the AWS Console Mobile Application? (p. 9)
- Can I use modifier keys on my IOS and Android keyboards when using AWS CloudShell in the AWS Console Mobile Application? (p. 10)
- Can I split the AWS CloudShell tab display into multiple tabs on the AWS Console Mobile Application? (p. 10)
- Can I access AWS CloudShell on the Console Toolbar on a mobile device? (p. 10)

How do I get started using AWS CloudShell?

You can get started by launching AWS CloudShell in a few steps from the AWS Management Console. To do this, sign in to the console using your AWS account or IAM credentials at https://console.aws.amazon.com/console/home.

For more information, see Getting started with AWS CloudShell (p. 11).

Back to list of FAQ (p. 6)

What do I need to access AWS CloudShell?

Because you access AWS CloudShell from the AWS Management Console, you must be an IAM user who can provide a valid account alias or ID, user name, and password.

To launch AWS CloudShell on the console, you need the IAM permissions that the attached policy provides. For more information, see Managing AWS CloudShell access and usage with IAM policies (p. 43).
What is AWS CloudShell on the Console Toolbar?

The CloudShell icon on the lower left of the AWS Management Console.

How do I launch AWS CloudShell on the Console Toolbar?

You can launch AWS CloudShell on the Console Toolbar by choosing the CloudShell icon on the lower left of the console.

Which AWS Regions is AWS CloudShell available in?

For a list of supported AWS Regions and the associated service endpoints, see the AWS CloudShell page in the Amazon Web Services General Reference.

Which AWS Region is assigned if AWS CloudShell isn't available in the selected Region when you launch CloudShell on the Console Toolbar?

The default Region is assigned to a Region that's closest to the selected Region. For more information, see Select a Region, launch AWS CloudShell, and choose a shell (p. 14).

You can run the command that provides permissions to manage resources in a different Region than the default Region. For more information, see Working in AWS Regions (p. 32).

What types of shells can I use in AWS CloudShell?

In AWS CloudShell, you can run commands using the Bash shell, PowerShell, or the Z shell. To switch shells, enter the shell name that you want to use using the following format on the command prompt:

• bash: Use the Bash shell
• pwsh: Use PowerShell
• zsh: Use the Z shell

What web browsers can I use with AWS CloudShell?

AWS CloudShell supports the four most recent versions of Google Chrome, Mozilla Firefox, Microsoft Edge, and Apple Safari.
Which web browsers can I use when I launch AWS CloudShell on the Console Toolbar?

You can launch CloudShell on the Console Toolbar using the most recent versions of Google Chrome and Microsoft Edge. Currently, you can't launch CloudShell on the Console Toolbar using Mozilla Firefox and Apple Safari.

Can I download a file when I launch AWS CloudShell on the Console Toolbar?

Yes, you can download a file when you launch CloudShell on the Console Toolbar. You can download a file using the most recent version of Google Chrome.

Currently, you can't download a file using Mozilla Firefox, Microsoft Edge, and Apple Safari.

What software is pre-installed on my shell environment?

With the shell that's created for AWS CloudShell sessions, you can switch seamlessly between your preferred command-line shells (Bash, PowerShell, and Z shell). You can also have access to pre-installed tools and utilities such as Make, pip, sudo, tar, tmux, Vim, Wget and Zip.

The shell environment is pre-configured with support for most major software languages. For example, you can use it to run Node.js and Python projects without first having to perform runtime installations. PowerShell users can use the .NET Core runtime.

You can add files that were created using the shell or uploaded with the shell interface to a version-controlled repository managed using a pre-installed version of git.

For more information, see Pre-installed software (p. 56).

Can I install software that's not available in the shell environment?

Yes, AWS CloudShell users have sudo privileges and can install software from the command line. For more information, see Installing third-party software on your shell environment (p. 64).

Can I restrict the actions that users can do in AWS CloudShell?

Yes, you can control what actions users can do in AWS CloudShell. For example, you can allow users to access AWS CloudShell but prevent them from uploading or downloading files within the shell...
How can I move data from my home directory if I want to change the AWS Region where I'm using AWS CloudShell?

To move your AWS CloudShell data from one AWS Region to another Region, first download the contents of your home directory in one Region to your local machine, next upload from it to the home directory in another Region. For more information, see Tutorial: Copying multiple files between your local machine and AWS CloudShell (p. 23).

Can I increase the limit that determines when AWS CloudShell times out because of user inactivity?

Your shell session automatically ends after approximately 20–30 minutes if you don’t interact with AWS CloudShell using your keyboard or pointer. Running processes don’t count as interactions. Because CloudShell is designed for focused, task-based activities, there are no plans at present to increase this timeout limit (p. 75).

If you want to perform terminal-based tasks using an AWS service with more flexible timeouts, we recommend using our cloud-based IDE, AWS Cloud9, or launching and connecting to an Amazon EC2 instance.

Can I access AWS CloudShell in the AWS Console Mobile Application from the home screen?

Yes, you can access AWS CloudShell in the AWS Console Mobile Application by logging in to the Console Mobile Application. For more information, see AWS Console Mobile Application User Guide.

How can I launch AWS CloudShell in the AWS Console Mobile Application?

You can launch AWS CloudShell using one of the following methods:

1. Select the AWS CloudShell icon at the bottom of the navigation bar.
2. Select the AWS CloudShell on the Services menu.
Can I use modifier keys on my iOS and Android keyboards when using AWS CloudShell in the AWS Console Mobile Application?

Yes, you can use modifier keys on your iOS and Android keyboards. For more information, see AWS Console Mobile Application User Guide.

Can I split the AWS CloudShell tab display into multiple tabs on the AWS Console Mobile Application?

No, currently you can't run multiple AWS CloudShell tabs on your mobile application.

Can I access AWS CloudShell on the Console Toolbar on a mobile device?

No, currently you can't access AWS CloudShell on the Console Toolbar on your mobile device.
Getting started with AWS CloudShell

This introductory tutorial shows you how to launch AWS CloudShell and perform key tasks using the shell command line interface.

First, you sign in to the AWS Management Console and select an AWS Region. You then launch CloudShell in a new browser window and a shell type to work with.

Next, you create a new folder in your home directory and upload a file to it from your local machine. You work on that file using a pre-installed editor before running it as a program from the command line. Last, you call AWS CLI commands to create an Amazon S3 bucket and add your file as an object to the bucket.

Prerequisites

IAM permissions

You can obtain permissions for AWS CloudShell by attaching the following AWS managed policy to your IAM identity (such as a user, role, or group):

- **AWSCloudShellFullAccess**: Provides users with full access to AWS CloudShell and its features.

For this tutorial, you also interact with AWS services. More specifically, you interact with Amazon S3 by creating an S3 bucket and adding an object to that bucket. Your IAM identity requires a policy that grants, at a minimum, the `s3:CreateBucket` and `s3:PutObject` permissions.

For more information, see Amazon S3 Actions in the Amazon Simple Storage Service User Guide.

Exercise file

This exercise also involves uploading and editing a file that's then run as a program from the command line interface. Open a text editor on your local machine and add the following code snippet.

```python
import sys
x=int(sys.argv[1])
y=int(sys.argv[2])
sum=x+y
print("The sum is",sum)
```

Save the file with the name `add_prog.py`.

Contents

- **Step 1**: Sign in to AWS Management Console (p. 12)
- **Step 2**: Select a Region, launch AWS CloudShell, and choose a shell (p. 14)
- **Step 3**: Download a file from AWS CloudShell (p. 15)
- **Step 4**: Upload a file to AWS CloudShell (p. 16)
- **Step 5**: Remove a file from AWS CloudShell (p. 17)
- **Step 6**: Create a home directory backup (p. 17)
- **Step 7**: Restart a shell session (p. 18)
Step 1: Sign in to AWS Management Console

This step involves entering your IAM user information to access the AWS Management Console. If you're already in the console, skip to step 2 (p. 14).

You can access the AWS Management Console by using an IAM users sign-in URL or going to the main sign-in page.

IAM user sign-in URL
- Open a browser and enter the following sign-in URL. Replace account_alias_or_id with the account alias or account ID that your administrator provided.

https://account_alias_or_id.signin.aws.amazon.com/console/
- Enter your IAM sign-in credentials and choose Sign in.

Sign in as IAM user

Account ID (12 digits) or account alias
- account_alias_or_id

IAM user name

Password

Sign in

Sign in using root user email

Forgot password?

Main sign-in page
- If you didn't sign in previously using this browser, the main sign-in page appears. Choose IAM user, enter the account alias or account ID, and choose Next.
Step 1: Sign in to AWS Management Console

• If you already signed in as an IAM user before. Your browser might remember the account alias or account ID for the AWS account. If so, enter your IAM sign-in credentials and choose Sign in.

Sign in as IAM user

Account ID (12 digits) or account alias

account_alias_or_id

IAM user name

Password

Sign in

Sign in using root user email

Forgot password?
Step 2: Select a Region, launch AWS CloudShell, and choose a shell

In this step, you launch AWS CloudShell from the console interface, choose an available AWS Region, and switch to your preferred shell, such as Bash, PowerShell, or Z shell.

1. To choose an AWS Region to work in, go to the Select a Region menu and select a supported AWS Region (p. 72) to work in. (Available Regions are highlighted.)

   **Important**
   If you switch Regions, the interface refreshes and the name of the selected AWS Region is displayed above the command line text. Any files that you add to persistent storage are available only in this same AWS Region. If you change Regions, different storage and files are accessible.

   **Important**
   If CloudShell isn't available in the selected Region when you launch CloudShell on the Console Toolbar, on the lower left of the console, then the default Region is set to a Region that's closest to the selected Region. You can run the command that provides permissions to manage resources in a different Region than the default Region. For more information, see Working in AWS Regions (p. 32).

   **Example**

   If you choose Europe (Spain) eu-south-2 but CloudShell isn't available in Europe (Spain) eu-south-2, then the default Region is set to Europe (Ireland) eu-west-1, which is closest to the Europe (Spain) eu-south-2.

   You will use the service quotas for the default Region, Europe (Ireland) eu-west-1 and the same CloudShell session will be restored across all Regions. The default Region might be changed and you will be notified in the CloudShell browser window.

2. From the AWS Management Console, you can launch CloudShell by choosing one of the following options:

   1. On the navigation bar, choose the CloudShell icon.
   2. In the Search box, type “CloudShell”, and then choose CloudShell.
   3. In the Recently visited widget, choose CloudShell.
   4. Choose CloudShell on the Console Toolbar, on the lower left of the console.
      - To adjust the height of your CloudShell session, drag =.
      - To switch your CloudShell session to a full screen, click Open in new browser tab icon.
When the command prompt displays, the shell is ready for interaction.

**Note**
If you encounter issues that prevent you from successfully launching or interacting with AWS CloudShell, check for information to identify and address those issues in Troubleshooting AWS CloudShell (p. 67).

3. To choose a pre-installed shell to work with, enter its program name at the command line prompt.

   Bash
   ```
   bash
   ```
   If you switch to Bash, the symbol at the command prompt updates to $.

   **Note**
   Bash is the default shell that's running when you launch AWS CloudShell.

   PowerShell
   ```
   pwsh
   ```
   If you switch to PowerShell, the symbol at the command prompt updates to PS>.

   Z shell
   ```
   zsh
   ```
   If you switch to Z shell, the symbol at the command prompt updates to %.

For information about the versions pre-installed in your shell environment, see the shells table (p. 56) in the AWS CloudShell compute environment (p. 55) section.

**Step 3: Download a file from AWS CloudShell**

This step walks you through the process of downloading a file.
1. To download a file, go to Actions and choose Download file from the menu.

   The Download file dialog box displays.

2. In the Download file dialog box, enter the path for the file to be downloaded.

   **Download file**

   Download files from your AWS CloudShell to your local desktop. Folders are not supported.

   ![Download file dialog box](image)

   **Individual file path**

   You can copy the file path from the command-line and paste it below.

   /home/cloudshell-user/subfolder/mydownloadfile.txt

   myfile.txt or /folder/myfile.txt.

   **Note**

   You can use absolute or relative paths when specifying a file for download. With relative pathnames, /home/cloudshell-user/ is added automatically to the start by default. So, to download a file called mydownloadfile, both of the following are valid paths:

   - **Absolute path:** /home/cloudshell-user/subfolder/mydownloadfile.txt
   - **Relative path:** subfolder/mydownloadfile.txt

3. Choose Download.

   If the file path is correct, a dialog box displays. You can use this dialog box to open the file with the default application. Or, you can save the file to a folder on your local machine.

   **Note**

   The Download option isn't available when you launch CloudShell on the Console Toolbar. You can download a file from CloudShell console or using the Chrome web browser. For more information about how to download a file, see Step 3: Download a file from AWS CloudShell (p. 15).

---

**Step 4: Upload a file to AWS CloudShell**

This step describes how to upload a file and then moving it to a new directory in your home directory.

1. To check your current working directory, at the prompt enter the following command:

   `pwd`

   When you press Enter, the shell returns your current working directory (for example, /home/cloudshell-user).

2. To upload a file to this directory, go to Actions and choose Upload file from the menu.

   The Upload file dialog box displays.

3. Choose Browse.
4. In your system's File upload dialog box, select the text file that you created for this tutorial (add_prog.py) and choose Open.

5. In the Upload file dialog box, choose Upload.

   A progress bar tracks the upload. If the upload is successful, a message confirms that add_prog.py was added to the root of your home directory.

6. To create a directory for the file, enter the make directories command: mkdir mysub_dir.

7. To move the uploaded file from the root of your home directory to the new directory, use the mv command:

   mv add_prog.py mysub_dir.

8. To change your working directory to the new directory, enter cd mysub_dir.

   The command prompt updates to indicate you've changed your working directory.

9. To view the contents of the current directory, mysub_dir, enter the ls command.

   The contents of the working directory are listed. This includes the file that you just uploaded.

---

**Step 5: Remove a file from AWS CloudShell**

This step describes how to remove a file from AWS CloudShell.

1. To remove a file from AWS CloudShell, use standard shell commands such as rm (remove).

   rm my-file-for-removal

2. To remove multiple files that meet specified criteria, run the find command.

   The following example removes all the files that include the suffix "pdf" in their names.

   ```bash
   find -type f -name '*.pdf' -delete
   ```

**Note**

Suppose that you stop using AWS CloudShell in a specific AWS Region. Then, the data that's in that Region's persistent storage is removed automatically after a specified period. For more information, see Persistent Storage (p. 74).

---

**Step 6: Create a home directory backup**

1. **Create a backup file**

   Create a temporary folder outside the home directory.

   ```bash
   HOME_BACKUP_DIR=$(mktemp --directory)
   ```

   You can use one of the following options to create a backup:

   a. **Create a backup file using tar**

      To create a backup file using tar, enter the following command:

      ```bash
      tar \\
      --create \\
      ```
Step 7: Restart a shell session

**Note**
As a security measure, if you don’t interact with the shell using the keyboard or pointer for an extended period, the session stops automatically. Long-running sessions are also automatically stopped. For more information, see [Shell sessions](p. 75).
1. To restart a shell session, choose **Actions, Restart AWS CloudShell**.
   You're notified that restarting AWS CloudShell stops all active sessions in the current AWS Region.

2. To confirm, choose **Restart**.
   An interface displays a message that the CloudShell compute environment is stopping. After the environment stopped and restarted, you can start working with the command line in a new session.
   
   **Note**
   In some cases, it may take a few minutes for your environment to restart.

---

### Step 8: Delete a shell session home directory

**Warning**
Deleting your home directory is an irreversible action where all the data that's stored in your home directory is deleted permanently. However, you might want to consider this option in the following situations:

- You incorrectly modified a file and can't access the AWS CloudShell compute environment. Deleting your home directory returns AWS CloudShell to its default settings.
- You want to remove all your data from AWS CloudShell immediately. If you stop using AWS CloudShell in an AWS Region, persistent storage is automatically deleted at the end of the retention period (p. 74) unless you launch AWS CloudShell again in the Region.

If you require long-term storage for your files, please consider a service such as Amazon S3 or CodeCommit.

1. To delete a shell session, choose **Actions, Delete AWS CloudShell home directory**.
   You're notified that deleting AWS CloudShell home directory deletes all data currently stored in your AWS CloudShell environment.
   
   **Note**
   You can't undo this action.

2. To confirm deletion, enter delete in the text input field, and then choose **Delete**.

   **Delete AWS CloudShell home directory**
   
   Deleting your home directory will delete all data currently stored in your AWS CloudShell environment. This action cannot be undone. AWS CloudShell stops all active sessions in the current AWS Region and creates a new environment immediately.

   To confirm deletion, enter **delete** in the text input field.

   ```
   delete
   ```

   AWS CloudShell stops all active sessions in the current AWS Region and creates a new environment immediately.

---

### Manually exiting shell sessions

AWS CloudShell allows you to manually exit session environments using command line options.
Step 9: Edit your file's code and run it using the command line

This step demonstrates how to use the pre-installed Vim editor to work with a file. You then run that file as a program from the command line.

1. To edit the file you uploaded in the previous step, enter the following command:

   ```
   vim add_prog.py
   ```

   The shell interface refreshes to display the Vim editor.

2. To edit the file in Vim, press the I key. Now edit the contents so the program adds up three numbers instead of two.

   ```python
   import sys
   x=int(sys.argv[1])
   y=int(sys.argv[2])
   z=int(sys.argv[3])
   sum=x+y+z
   print("The sum is",sum)
   ```

   **Note**
   
   If you paste the text into the editor and have the Safe Paste feature (p. 39) enabled, a warning is displayed. Multiline text that's copied can contain malicious scripts. With the Safe Paste feature, you can verify the complete text before it's pasted in. If you're satisfied that the text is safe, choose Paste.

3. After you edited the program, press Esc to enter the Vim command mode. Then, enter the :wq command to save the file and exit the editor.

   **Note**
   
   If you're new to the Vim command mode, you might initially find it challenging to switch between command mode and insert mode. Command mode is used when saving files and exiting the application. Insert mode is used when inserting new text. To enter insert mode, press I, and, to enter command mode, press Esc. For more information about Vim and other tools that are available in AWS CloudShell, see Development tools and shell utilities (p. 59).

4. On the main command line interface, run the following program and specify three numbers for input. The syntax is as follows.

   ```
   python3 add_prog.py 4 5 6
   ```

   The command line displays the program output: The sum is 15.
Step 10: Use AWS CLI to add the file as an object in an Amazon S3 bucket

In this step, you create an Amazon S3 bucket and then use the **PutObject** method to add your code file as an object in that bucket.

**Note**
In most cases, you can use a service such as CodeCommit (p. 26) to commit a software file into a version-controlled repository. This tutorial shows how you can use AWS CLI in AWS CloudShell to interact with other AWS services. Using this method, you don't need to download or install any additional resource. Moreover, because you're already authenticated within the shell, you don't need to configure credentials before making calls.

1. To create a bucket in a specified AWS Region, enter the following command:

   ```bash
   aws s3api create-bucket --bucket insert-unique-bucket-name-here --region us-east-1
   ``

   **Note**
   If you're creating a bucket outside of the us-east-1 Region, add `create-bucket-configuration` with the `LocationConstraint` parameter to specify the Region. The following is example syntax.

   ```bash
   $ aws s3api create-bucket --bucket my-bucket --region eu-west-1 --create-bucket-configuration LocationConstraint=eu-west-1
   ```

   If the call is successful, the command line displays a response from the service similar to the following output.

   ```json
   { 
     "Location": "/insert-unique-bucket-name-here"
   }
   ```

   **Note**
   If you don't adhere to the rules for naming buckets, the following error is displayed: An error occurred (InvalidBucketName) when calling the CreateBucket operation: The specified bucket is not valid.

2. To upload a file and add the file as an object to the bucket that you just created, call the **PutObject** method.

   ```bash
   aws s3api put-object --bucket insert-unique-bucket-name-here --key add_prog --body add_prog.py
   ```

   After the object is uploaded to the Amazon S3 bucket, the command line displays a response from the service similar to the following output:

   ```json
   {"ETag": "\"ab123c1:w:wad4a567d8bfd9a1234ebeea56\""}
   ```

   The ETag is the hash of the object that was stored. You can use this hash to check the integrity of the object uploaded to Amazon S3.
Related topics

- Working with AWS services in AWS CloudShell (p. 35)
- Tutorial: Copying multiple files between your local machine and AWS CloudShell (p. 23)
- Tutorial: Using CodeCommit in AWS CloudShell (p. 26)
- Working with AWS CloudShell (p. 31)
- Customizing your AWS CloudShell experience (p. 38)
AWS CloudShell tutorials

The following tutorials shows how to copy multiple files between your local machine and CloudShell, how to create a CodeCommit repository and clone it to your CloudShell compute environment, and how to create a presigned URL for Amazon S3 objects using CloudShell.

Topics

• Tutorial: Copying multiple files between your local machine and AWS CloudShell (p. 23)
• Tutorial: Using CodeCommit in AWS CloudShell (p. 26)
• Tutorial: Creating a presigned URL for Amazon S3 objects using AWS CloudShell (p. 28)

Tutorial: Copying multiple files between your local machine and AWS CloudShell

Using the CloudShell interface, you can upload or download a single file between your local machine and the shell environment at a time. To copy multiple files between CloudShell and your local machine at the same time, use one of the following options:

• Amazon S3: Use S3 buckets as an intermediary when copying files between your local machine and CloudShell.
• Zip files: Compress multiple files in a single zipped folder that can be uploaded or downloaded using the CloudShell interface.

Note

Because CloudShell doesn't allow incoming internet traffic, it's currently not possible to use commands such as scp or rsync to copy multiple files between local machines and the CloudShell compute environment.

Uploading and downloading multiple files using Amazon S3

Prerequisites

To work with buckets and objects, you need an IAM policy that grants permissions to perform the following Amazon S3 API actions:

- s3:CreateBucket
- s3:PutObject
- s3:GetObject

For a complete list of Amazon S3 actions, see Actions in the Amazon Simple Storage Service API Reference.

Upload multiple files to AWS CloudShell using Amazon S3

1. In AWS CloudShell, create an S3 bucket by running the following s3 command:
aws s3api create-bucket --bucket your-bucket-name --region us-east-1

If the call is successful, the command line displays a response from the S3 service:

```json
{
   "Location": "/your-bucket-name"
}
```

2. Upload the files in a directory from your local machine to the bucket. Choose one of the following options to upload files:

- AWS Management Console: Use drag-and-drop to upload files and folders to a bucket.
- AWS CLI: With the version of the tool installed on your local machine, use the command line to upload files and folders to the bucket.

Using the console

- Open the Amazon S3 console at [https://s3.console.aws.amazon.com/s3/](https://s3.console.aws.amazon.com/s3/).
  (If you're using AWS CloudShell, you should already be logged in to the console.)
- In the left navigation pane, choose Buckets, and then choose the name of the bucket that you want to upload your folders or files to. You can also create a bucket of your choice by choosing Create bucket.
- To select the files and folders that you want to upload, choose Upload. Then, drag and drop your selected files and folders into the console window that lists the objects in the destination bucket, or choose Add files, or Add folders.
  
  The files you chose are listed on the Upload page.
- Select the check boxes to indicate the files to be added.
- To add the selected files to the bucket, choose Upload.

**Note**

For information about the full range of configuration options when using the console, see [How do I upload files and folders to an S3 bucket?](https://docs.aws.amazon.com/AmazonS3/latest/userguide/uploading-files.html) in the Amazon Simple Storage Service User Guide.

Using AWS CLI

**Note**

For this option, you need to have the AWS CLI tool installed on your local machine and have your credentials configured for calls to AWS services. For more information, see the [AWS Command Line Interface User Guide](https://docs.aws.amazon.com/cli/latest/userguide/).

- Launch the AWS CLI tool and run the following aws s3 command to sync the specified bucket with the contents of the current directory on your local machine:

  ```bash
  aws s3 sync folder-path s3://your-bucket-name
  ```

  If the sync is successful, upload messages are displayed for every object added to the bucket.

3. Return to the CloudShell command line and enter the following command to synchronize the directory in the shell environment with the contents of the S3 bucket:
aws s3 sync s3://your-bucket-name folder-path

Note
You can also add --exclude "<value>" and --include "<value>" parameters to the sync command to perform pattern matching to either exclude or include a particular file or object. For more information, see Use of Exclude and Include Filters in the AWS CLI Command Reference.

If the sync is successful, upload messages are displayed for every object added to the bucket.

Note
With the sync command, only new and updated files are recursively copied from the source directory to the destination.

Download multiple files from AWS CloudShell using Amazon S3

1. Using the AWS CloudShell command line, enter the following aws s3 command to sync an S3 bucket with contents of the current directory in the shell environment:

   aws s3 sync folder-path s3://your-bucket-name

   Note
   You can also add --exclude "<value>" and --include "<value>" parameters to the sync command to perform pattern matching to either exclude or include a particular file or object. For more information, see Use of Exclude and Include Filters in the AWS CLI Command Reference.

   If the sync is successful, upload messages are displayed for every object added to the bucket.

2. Download the contents of the bucket to your local machine. Because the Amazon S3 console doesn't support the downloading of multiple objects, you need to use the AWS CLI tool that's installed on your local machine.

   From the command line of the AWS CLI tool, run the following command:

   aws s3 sync s3://your-bucket-name folder-path

   If the sync is successful, the command line displays a download message for each file updated or added in the destination directory.

   Note
   For this option, you need to have the AWS CLI tool installed on your local machine and have your credentials configured for calls to AWS services. For more information, see the AWS Command Line Interface User Guide.

Uploading and downloading multiple files using zipped folders

With the zip/unzip utilities, you can compress multiple files in an archive that can be treated as a single file. The utilities are pre-installed in the CloudShell compute environment.
For more information about pre-installed tools, see Development tools and shell utilities (p. 59).

**Upload multiple files to AWS CloudShell using zipped folders**

1. On your local machine, add the files to be uploaded to a zipped folder.
2. Launch CloudShell, and then choose Actions, Upload file.
3. In the Upload file dialog box, choose Select file, and then choose the zipped folder you just created.
4. In the Upload file dialog box, choose Upload to add the selected file to the shell environment.
5. In the CloudShell command line, run the following command to unzip the contents of the zip archive to a specified directory:

   ```
   unzip zipped-files.zip -d my-unzipped-folder
   ```

**Download multiple files from AWS CloudShell using zipped folders**

1. In the CloudShell command line, running the following command to add all the files in the current directory to a zipped folder:

   ```
   zip -r zipped-archive.zip *
   ```

2. Choose Actions, Download file.
3. In the Download file dialog box, enter the path for the zipped folder (``/home/cloudshell-user/zip-folder/zipped-archive.zip``, for example), and then choose Download.

   If the path is correct, a browser dialog offers the choice of opening the zipped folder or saving it to your local machine.

4. On your local machine, you can now unzip the contents of the downloaded zipped folder.

---

**Tutorial: Using CodeCommit in AWS CloudShell**

CodeCommit is a secure, highly scalable, and managed source control service that hosts private Git repositories. Using AWS CloudShell, you can work with CodeCommit on the command line using the `git-remote-codecommit` utility. This utility is pre-installed in the AWS CloudShell compute environment and provides a simple method for pushing and pulling code from CodeCommit repositories. This utility does this by extending Git. For more information, see the AWS CodeCommit User Guide.

This tutorial describes how to create a CodeCommit repository and clone it to your AWS CloudShell compute environment. You also learn how to stage and commit a file to your cloned repository before pushing it to the remote repository that's managed in AWS Cloud.

**Prerequisites**

For information about the permissions that an IAM user requires to use AWS CloudShell, see the prerequisites section in the Getting started tutorial (p. 11). You also need IAM permissions to work with CodeCommit.

In addition, before starting, make sure to have the following:

- A basic understanding of Git commands and version control concepts
- A file in the home directory of your shell that can be committed to the local and remote repositories. In this tutorial, it's referred to as my-git-file.
Step 1: Create and clone a CodeCommit repository

1. In the CloudShell command line interface, enter the following `codecommit` command to create a CodeCommit repository called MyDemoRepo.

```
aws codecommit create-repository --repository-name MyDemoRepo --repository-description "My demonstration repository"
```

If the repository is successfully created, the command line displays the service's response.

```
{
    "repositoryMetadata": {
        "accountId": "111111111111",
        "repositoryId": "0dcd29a8-941a-1111-1111-11111111111a",
        "repositoryName": "MyDemoRepo",
        "repositoryDescription": "My demonstration repository",
        "lastModifiedDate": "2020-11-23T20:38:23.068000+00:00",
        "creationDate": "2020-11-23T20:38:23.068000+00:00",
        "cloneUrlHttp": "https://git-codecommit.eu-west-1.amazonaws.com/v1/repos/MyDemoRepo",
        "cloneUrlSsh": "ssh://git-codecommit.eu-west-1.amazonaws.com/v1/repos/MyDemoRepo",
        "Arn": "arn:aws:codecommit:eu-west-1:111111111111:MyDemoRepo"
    }
}
```

2. Using the command line, create a new directory for your local repository and make it your working directory.

```
mkdir my-shell-repo
cd my-shell-repo
```

3. To clone the remote repository, use the `git clone` command. (As you're working with `git-remote-codecommit`, use the HTTPS (GRC) URL style).

```
git clone codecommit::eu-west-1://MyDemoRepo
```

If the repository is successfully cloned, the command line displays the service's response.

```
Cloning into 'MyDemoRepo'...
warning: You appear to have cloned an empty repository.
```

4. To navigate to the cloned repository, use the `cd` command.

```
cd MyDemoRepo
```

Step 2: Stage and commit a file before pushing it to your CodeCommit repository

1. Add a file called `my-git-file` to the MyDemoRepo folder using either a Vim editor or the file upload feature of AWS CloudShell. To learn how to use both, see the Getting started tutorial (p. 11).

2. To stage your file in the repository, run the `git add` command.

```
git add my-git-file
```
3. To check that the file has been staged and is ready to be committed, run the `git status` command.

```
git status
```

`my-git-file` is listed as a new file and displays in green text, indicating it's ready to be committed.

4. Commit this version of the staged file to the repository.

```
git commit -m "first commit to repo"
```

**Note**
If you're asked for configuration information to complete the commit, use the following format.

```
$ git config --global user.name "Jane Doe"
$ git config --global user.email janedoe@example.com
```

5. To sync your remote repository with the changes made in your local one, push the changes to the upstream branch.

```
git push
```

---

**Tutorial: Creating a presigned URL for Amazon S3 objects using AWS CloudShell**

This tutorial shows you how to create a presigned URL to share an Amazon S3 object with others. Because object owners specify their own security credentials when sharing, anyone who receives the presigned URL can access the object for a limited time.

**Prerequisites**

- An IAM user with access permissions provided by the **AWSCloudShellFullAccess** policy.
- For the IAM permissions that are required to create a presigned URL, see [Share an object with others](https://docs.aws.amazon.com/AmazonS3/latest/userguide/share-object-with-others.html) in the **Amazon Simple Storage Service User Guide**.

**Step 1: Create an IAM role to grant access to Amazon S3 bucket**

1. To get your IAM details that can be shared, call the `get-caller-identity` command from AWS CloudShell.

```
aws sts get-caller-identity
```

If the call is successful, the command line displays a response similar to the following.

```
"Account": "123456789012",
"UserId": "AROAXXXOZUOOTTWDCVIDZ2:redirect_session",
"Arn": "arn:aws:sts::531421766567:assumed-role/Feder08/redirect_session"
}

2. Take the user information that you obtained in the previous step, and add it to an AWS CloudFormation template. This template creates an IAM role. This role grants your collaborator least-privilege permissions for the shared resources.

Resources:
  CollaboratorRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
        - Effect: Allow
          Principal:
            AWS: "arn:aws:iam::531421766567:role/Feder08"
          Action: "sts:AssumeRole"
          Description: Role used by my collaborators
          MaxSessionDuration: 7200
        CollaboratorPolicy:
          Type: AWS::IAM::Policy
          Properties:
            PolicyDocument:
              Version: 2012-10-17
              Statement:
              - Effect: Allow
                Action:
                - 's3:*'
                Resource: 'arn:aws:s3:::<YOUR_BUCKET_FOR_FILE_TRANSFER>'
                Condition:
                  StringEquals:
                    s3:prefix:
                    - "myfolder/**"
          PolicyName: S3ReadSpecificFolder
          Roles:
          - !Ref CollaboratorRole
        Outputs:
          CollaboratorRoleArn:
            Description: Arn for the Collaborator's Role
            Value: !GetAtt CollaboratorRole.Arn

3. Save the AWS CloudFormation template in a file that's named template.yaml.

4. Use the template to deploy the stack and create the IAM role by calling the deploy command.

```
aws cloudformation deploy --template-file ./template.yaml --stack-name CollaboratorRole
--capabilities CAPABILITY_IAM
```

Generate the presigned URL

1. Using your editor in AWS CloudShell, add the following code. This code creates a URL that provides federated users with direct access to the AWS Management Console.

```
import urllib, json, sys
import requests
import boto3
import os
```
def main():
    sts_client = boto3.client('sts')
    assume_role_response = sts_client.assume_role(
        RoleArn=os.environ.get(ROLE_ARN),
        RoleSessionName="collaborator-session"
    )
    credentials = assume_role_response["Credentials"]
    url_credentials = {}
    url_credentials["sessionId"] = credentials.get("AccessKeyId")
    url_credentials["sessionKey"] = credentials.get("SecretAccessKey")
    url_credentials["sessionToken"] = credentials.get("SessionToken")
    json_string_with_temp_credentials = json.dumps(url_credentials)
    print(f"json string {json_string_with_temp_credentials}"

    request_parameters = f"?Action=getSigninToken&Session={urllib.parse.quote(json_string_with_temp_credentials)}"
    request_url = "https://signin.aws.amazon.com/federation" + request_parameters
    r = requests.get(request_url)
    signin_token = json.loads(r.text)
    request_parameters = f"Action=login&Issuer=Example.org&Destination={urllib.parse.quote("https://us-west-2.console.aws.amazon.com/cloudshell")}"
    request_url = "https://signin.aws.amazon.com/federation" + request_parameters

    # Send final URL to stdout
    print (request_url)

if __name__ == "__main__":
    main()
Working with AWS CloudShell

This topic describes how you can navigate the shell interface, work in AWS Region, and work with files to upload and download from the shell environment.

Navigating the AWS CloudShell interface

You can navigate CloudShell interface features from the AWS Management Console and Console Toolbar. The following screenshot indicates several key AWS CloudShell interface features.

1. AWS CloudShell command line interface that you use to run commands by using your preferred shell (p. 14). The current shell type is indicated by the command prompt.
2. The terminal tab, which uses AWS Region where AWS CloudShell is currently running.
3. The Actions menu, which provides options for changing the screen layout (p. 38), downloading (p. 15) and uploading (p. 16) files, restarting your AWS CloudShell (p. 18), and deleting your AWS CloudShell home directory (p. 19).

   Note
   The Download option isn't available when you launch CloudShell on the Console Toolbar.

4. The Open in new browser tab, which provides the option to access your CloudShell session in a full screen.
5. The Preferences option, which you can use to customize your shell experience (p. 38).
6. The bottom bar, which provides the following options to:
   - Launch CloudShell from the CloudShell icon.
   - Provide feedback from the Feedback icon. Choose the type of feedback that you want to submit, add your comments, and then choose Submit.
   - To submit feedback for CloudShell, choose one of the following options:
Working in AWS Regions

The current AWS Region that you're running in is displayed above the command line interface.

You can choose an AWS Region to work in by selecting a specific Region using the Region selector. After you change Regions, the interface refreshes as your shell session connects to a different compute environment that's running in the selected Region.

Important
You can use up to 1 GB of persistent storage in each AWS Region. Persistent storage is stored in your home directory ($HOME). This means that any personal files, directories, programs, or scripts that are stored in your home directory are all located in one AWS Region. Moreover, they're different from those that are located in the home directory and stored a different Region.

The long-term retention of files in persistent storage is also managed on a per-Region basis. For more information, see Persistent storage (p. 74).

Specifying your default AWS Region for AWS CLI

You can use environment variables to specify configuration options and credentials required to access AWS services using AWS CLI. The environment variable that specifies the default AWS Region for your shell session is set in either when you launch AWS CloudShell from a specific Region in the AWS Management Console or when you choose an option in the Region selector.
Environment variables have precedence over AWS CLI credentials files that are updated by `aws configure`. So, you can't run the `aws configure` command to change the Region that's specified by the environment variable. Instead, to change the default Region for AWS CLI commands, assign a value to the `AWS_REGION` environment variable. In the examples that follow, replace `us-east-1` with the Region that you're in.

Bash or Zsh

```bash
$ export AWS_REGION=us-east-1
```

Setting the environment variable changes the value that's used until either at the end of your shell session or when you set the variable to a different value. You can set variables in your shell's startup script to make the variables persistent across future sessions.

PowerShell

```powershell
PS C:\> $Env:AWS_REGION="us-east-1"
```

If you set an environment variable at the PowerShell prompt, the environment variable saves the value for only the duration of the current session. Alternatively, you can set the variable for all future PowerShell sessions by adding the variable to your PowerShell profile. For more information about storing environment variables, see the PowerShell documentation.

To confirm that you've changed the default Region, run the `aws configure list` command to display the current AWS CLI configuration data.

**Note**
For specific AWS CLI commands, you can override the default Region using the command line option `--region`. For more information, see Command line options in the AWS Command Line Interface User Guide.

## Working with files and storage

Using AWS CloudShell's interface, you can upload files to and download files from the shell environment. For more information about downloading and uploading files, see Getting started with AWS CloudShell. (p. 11)

To ensure any of the files you add are available after your session ends, you should know the difference between persistent and temporary storage.

- **Persistent storage**: You have 1 GB of persistent storage for each AWS Region. Persistent storage is in your home directory.
- **Temporary storage**: Temporary storage is recycled at the end of a session. Temporary storage is in the directories that are outside your home directory.

**Important**
Make sure to leave files that you want to keep and use for future shell sessions in your home directory. For example, suppose that you move a file out of your home directory by running the `mv` command. Then, that file is recycled when the current shell session ends.
Working with accessibility features for AWS CloudShell

This topic describes how to use accessibility features for CloudShell. You can use a keyboard to navigate through the focusable elements on the page. You can also customize the appearance of CloudShell, including font sizes and interface themes.

Keyboard navigation in CloudShell

To navigate through the focusable elements on the page, press Tab.

CloudShell terminal accessibility features

You can use the Tab key in the following modes:

- **Terminal mode (Default)** – In this mode, the terminal captures your Tab key entry. After the focus is on the terminal, press Tab to access only the functionality of the terminal.
- **Navigation mode** – In this mode, the terminal doesn't capture your Tab key entry. Press Tab to navigate through the focusable elements on the page.

To switch between terminal mode and navigation mode, press Ctrl+M. After you switch back, Tab: navigation appears in the header, and you can use the Tab key to navigate through the page.

To return to terminal mode, press Ctrl+M. Or, choose X next to Tab: navigation.

Note
Currently, CloudShell terminal accessibility features are not available on mobile devices.

Choosing font sizes and interface themes in CloudShell

You can customize the appearance of CloudShell to accommodate your visual preferences.

- **Font size** – Choose from Smallest, Small, Medium, Large, and Largest font sizes in the terminal. For more information about changing the font size, see the section called “Changing font size” (p. 38).
- **Theme** – Choose between Light and Dark interface themes. For more information about changing the interface theme, see the section called “Changing the interface theme” (p. 39).
Working with AWS services in AWS CloudShell

A key benefit of AWS CloudShell is that you can use it to manage your AWS services from the command line interface. This means that you don't need to download and install tools or configure your credentials locally beforehand. When you launch AWS CloudShell, a compute environment is created that has the following AWS command line tools already installed:

- AWS CLI (p. 35)
- AWS Elastic Beanstalk CLI (p. 36)
- Amazon ECS CLI (p. 37)
- AWS SAM (p. 37)

And because you've already signed into AWS, there's no requirement to configure your credentials locally before using services. The credentials you used to sign in to the AWS Management Console are forwarded to AWS CloudShell.

If you want to change the default AWS Region used for AWS CLI, you can change the value assigned to the AWS_REGION environment variable. (For more information, see Specifying your default AWS Region for AWS CLI (p. 32).)

The rest of this topic demonstrates how you can start using AWS CloudShell to interact with selected AWS services from the command line.

AWS CLI command line examples for selected AWS services

The following examples represent only some of the numerous AWS services that you can work with using commands available from AWS CLI Version 2. For a full listing, see the AWS CLI Command Reference.

- DynamoDB (p. 35)
- AWS Cloud9 (p. 36)
- Amazon EC2 (p. 36)
- S3 Glacier (p. 36)

DynamoDB

DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. This service's implementation of the NoSQL mode supports key-value and document data structures.

The following create-table command creates a NoSQL-style table that's named MusicCollection in your AWS account.

```
aws dynamodb create-table \
  --table-name MusicCollection \
  --attribute-definitions AttributeName=Artist,AttributeType=S \
  AttributeName=SongTitle,AttributeType=S
```
AWS CloudShell User Guide

AWS Cloud9

--key-schema AttributeName=Artist,KeyType=HASH AttributeName=SongTitle,KeyType=RANGE 
--provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5 
--tags Key=Owner,Value=blueTeam

For more information, see Using DynamoDB with the AWS CLI in the AWS Command Line Interface User Guide.

AWS Cloud9

AWS Cloud9 is a cloud-based integrated development environment (IDE) that you can use to write, run, and debug your code in a browser window. The environment features a code editor, debugger, and terminal.

The following create-environment-ec2 command creates an AWS Cloud9 EC2 development environment with the specified settings. It launches an Amazon EC2 instance, and then connects from the instance to the environment.

```bash
aws cloud9 create-environment-ec2 --name my-demo-env --description "My demonstration development environment." --instance-type t2.micro --subnet-id subnet-1fab8aEX --automatic-stop-time-minutes 60 --owner-arn arn:aws:iam::123456789012:user/MyDemoUser
```

For more information, see AWS Cloud9 command-line reference.

Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure and resizable compute capacity in the cloud. It's designed to make web-scale cloud computing easier and more accessible.

The following run-instances command launches a t2.micro instance in the specified subnet of a VPC:

```bash
aws ec2 run-instances --image-id ami-xxxxxxxx --count 1 --instance-type t2.micro --key-name MyKeyPair --security-group-ids sg-903004f8 --subnet-id subnet-6e7f829e
```

For more information, see Using Amazon EC2 with the AWS CLI in the AWS Command Line Interface User Guide.

S3 Glacier

S3 Glacier and S3 Glacier Deep Archive are a secure, durable, and extremely low-cost Amazon S3 cloud storage classes for data archiving and long-term backup.

The following create-vault command creates a vault—a container for storing archives:

```bash
aws glacier create-vault --vault-name my-vault --account-id -
```

For more information, see Using Amazon S3 Glacier with the AWS CLI in the AWS Command Line Interface User Guide.

AWS Elastic Beanstalk CLI

The AWS Elastic Beanstalk CLI provides a command line interface made to simplify creating, updating, and monitoring environments from a local repository. In this context, an environment refers to a collection of AWS resources running an application version.
The following `create` command creates a new environment in a custom Amazon Virtual Private Cloud (VPC).

```
$ eb create dev-vpc --vpc.id vpc-0ce8dd99 --vpc.elbsubnets subnet-b356d7c6,subnet-02f74b0c --vpc.ec2subnets subnet-0bb7f0cd,subnet-3b6697c1 --vpc.securitygroup sg-70cff265
```

For more information, see the [EB CLI command reference](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/eb-cli-reference.html) in the *AWS Elastic Beanstalk Developer Guide*.

### Amazon ECS CLI

The Amazon Elastic Container Service (Amazon ECS) command line interface (CLI) provides several high-level commands. These are designed to simplify the processes of creating, updating, and monitoring clusters and tasks from a local development environment. (An Amazon ECS cluster is a logical grouping of tasks or services.)

The following `configure` command configures the Amazon ECS CLI to create a cluster configuration named `ecs-cli-demo`. This cluster configuration uses FARGATE as the default launch type for the `ecs-cli-demo` cluster in the `us-east-1` region.

```
ecs-cli configure --region us-east-1 --cluster ecs-cli-demo --default-launch-type FARGATE --config-name ecs-cli-demo
```

For more information, see the [Amazon ECS Command Line Reference](https://docs.aws.amazon.com/ecs/latest/userguide/cli-reference.html) in the *Amazon Elastic Container Service Developer Guide*.

### AWS SAM CLI

AWS SAM CLI is a command line tool that operates on an AWS Serverless Application Model template and application code. You can perform several tasks using it. These include invoking Lambda functions locally, creating a deployment package for your serverless application, and deploying your serverless application to the AWS Cloud.

The following `init` command initializes a new SAM project with required parameters passed as parameters:

```
sam init --runtime python3.7 --dependency-manager pip --app-template hello-world --name sam-app
```

For more information, see the [AWS SAM CLI command reference](https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/cli-reference.html) in the *AWS Serverless Application Model Developer Guide*. 
Customizing your AWS CloudShell experience

You can customize the following aspects of your AWS CloudShell experience:

- **Tabs layout (p. 38)**: Split the command line interface into multiple columns and rows.
- **Font size (p. 38)**: Adjust the size of the command line text.
- **Color theme (p. 39)**: Toggle between light and dark theme.
- **Safe Paste (p. 39)**: Switch a feature on or off that requires you to verify multiline text before it's pasted.
- **Tmux to session restore (p. 39)**: Using tmux restores your session until the session becomes inactive.

You can also extend your shell environment by installing your own software (p. 64) and modifying start-up shell scripts (p. 64).

Splitting the command line display into multiple tabs

Run multiple commands by splitting your command line interface into several panes.

**Note**

After opening multiple tabs, you can select one that you want to work in by clicking anywhere in the pane of your choosing. You can close a tab by choosing the x symbol, which is next to the Region name.

- Choose **Actions** and one of the following options from **Tabs layout**:
  - **New tab**: Add a new tab that's next to the currently active one.
  - **Split into rows**: Add a new tab in a row that's below the currently active one.
  - **Split into columns**: Add a new tab in a column that's next to the currently active one.

If there's not enough space to completely display each tab, scroll to see the entire tab. You can also select the split bars that separate panes and drag them by using the pointer to increase or reduce the pane size.

Changing font size

Increase or decrease the size of the text that's displayed in the command line interface.

1. To change the AWS CloudShell terminal settings, go to **Settings, Preferences**.
2. Choose a text size. Your options are **Smallest, Small, Medium, Large**, and **Largest**.
Changing the interface theme

Toggle between light and dark theme for the command line interface.

1. To change the AWS CloudShell theme, go to Settings, Preferences.
2. Choose Light or Dark.

Using Safe Paste for multiline text

Safe Paste is a security feature that prompts you to verify that the multiline text that you’re about to paste into the shell doesn't contain malicious scripts. Text that's copied from third-party sites can contain hidden code that triggers unexpected behaviors in your shell environment.

The Safe Paste dialog displays the complete text that you copied to your clipboard. If you’re satisfied that there’s no security risk, choose Paste.

**Warning: Pasting multiline text into AWS CloudShell**

Text that's copied from external sources can contain malicious scripts. Verify the text below before pasting.

```python
import sys
x=int(sys.argv[1])
y=int(sys.argv[2])
z=int(sys.argv[3])
total=x+y+z
print("The total is",total)
```

- Always ask before pasting multiline code

We recommend that you enable Safe Paste to catch potential security risks in scripts. You can switch this feature on or off by choosing Preferences, Enable Safe Paste and Disable Safe Paste.

Using tmux to session restore

AWS CloudShell uses tmux to restore the sessions across single or multiple browser tabs. If you refresh the browser tabs, it resumes your session until the session becomes inactive. For more information, see Session restore (p. 3).
Security for AWS CloudShell

Cloud security at Amazon Web Services (AWS) is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations. Security is a shared responsibility between AWS and you. The Shared Responsibility Model describes this as Security of the Cloud and Security in the Cloud.

Security of the Cloud – AWS is responsible for protecting the infrastructure that runs all of the services offered in the AWS Cloud and providing you with services that you can use securely. Our security responsibility is the highest priority at AWS, and the effectiveness of our security is regularly tested and verified by third-party auditors as part of the AWS Compliance Programs.

Security in the Cloud – Your responsibility is determined by the AWS service you are using, and other factors including the sensitivity of your data, your organization's requirements, and applicable laws and regulations.

AWS CloudShell follows the shared responsibility model through the specific AWS services it supports. For AWS service security information, see the AWS service security documentation page and AWS services that are in scope of AWS compliance efforts by compliance program.

The following topics show you how to configure AWS CloudShell to meet your security and compliance objectives.

Topics
- Data protection in AWS CloudShell (p. 40)
- Identity and access management in AWS CloudShell (p. 41)
- Logging and monitoring in AWS CloudShell (p. 47)
- Compliance validation for AWS CloudShell (p. 48)
- Resilience in AWS CloudShell (p. 51)
- Infrastructure security in AWS CloudShell (p. 52)
- Configuration and vulnerability analysis in AWS CloudShell (p. 52)
- Security best practices for AWS CloudShell (p. 52)
- AWS CloudShell Security FAQs (p. 53)

Data protection in AWS CloudShell

The AWS shared responsibility model applies to data protection in AWS CloudShell. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. This content includes the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the AWS Security Blog.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
Data encryption

• Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.

• If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

We strongly recommend that you never put confidential or sensitive information, such as your customers’ email addresses, into tags or free-form text fields such as a Name field. This includes when you work with AWS CloudShell or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Data encryption

Data encryption refers to protecting data when at rest (while stored in AWS CloudShell) and when in transit (as it travels between AWS CloudShell and service endpoints).

Encryption at rest using AWS KMS

Encryption at rest refers to protecting your data from unauthorized access by encrypting data while stored. When using AWS CloudShell, you have persistent storage of 1 GB per AWS Region at no cost. Persistent storage is located in your home directory ($HOME) and is private to you. Unlike ephemeral environment resources that are recycled after each shell session ends, data in your home directory persists.

The encryption of data stored in AWS CloudShell is implemented using cryptographic keys provided by AWS Key Management Service (AWS KMS). This is a managed AWS service for creating and controlling customer master keys (CMKs)—the encryption keys used to encrypt customer data that’s stored in the AWS CloudShell environment. AWS CloudShell generates and manages cryptographic keys for encrypting data on behalf of customers.

Encryption in transit

Encryption in transit refers to protecting your data from being intercepted while it moves between communication endpoints.

By default, all data communication between the client's web browser computer and the cloud-based AWS CloudShell is encrypted by sending everything through an HTTPS/TLS connection.

You don't need to do anything to enable the use of HTTPS/TLS for communication.

Identity and access management in AWS CloudShell

AWS Identity and Access Management (IAM) is an Amazon Web Services (AWS) service that helps an administrator securely control access to AWS resources. IAM administrators control who can be authenticated (signed in) and authorized (have permissions) to use resources in AWS services. IAM is an AWS service that you can use with no additional charge.

To use AWS CloudShell to access AWS, you need an AWS account and AWS credentials. To increase the security of your AWS account, consider using an IAM user to provide access credentials instead of your own AWS account credentials.
Audience

How you use AWS Identity and Access Management (IAM) differs, depending on what you do in AWS CloudShell.

Service user - If you use the AWS CloudShell service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more AWS CloudShell features, you require more permissions. Understand how access is managed so you know how to request the right permissions from your administrator. If you can't access a feature in AWS CloudShell, see Troubleshooting AWS CloudShell (p. 67).

Service administrator - If you're in charge of AWS CloudShell resources at your company, you probably have full access to AWS CloudShell. It's your job to determine which AWS CloudShell features and resources your team access. You submit requests to your IAM administrator to change the permissions of your service users. Review the information about this page to understand IAM concepts and terms.

IAM administrator - If you're an IAM administrator, consider learning the details about how you can write policies to manage IAM user access to services. For information that's specific to AWS CloudShell, see Managing AWS CloudShell access and usage with IAM policies (p. 43).

Authentication

You can access AWS as any of the following types of identities:

• AWS account root user – When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account root user and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the IAM User Guide.

• IAM user – An IAM user is an identity within your AWS account that has specific custom permissions (for example, permissions to create in AWS CloudShell). You can use an IAM sign-in credentials to sign in to secure AWS webpages like the AWS Management Console, AWS Discussion Forums, or the AWS Support Center.

In addition to sign-in credentials, you can also generate access keys for each user. You can use these keys when you access AWS services either through one of the several SDKs or by using the AWS
Command Line Interface (CLI). The SDK and CLI tools use the access keys to cryptographically sign your request. If you don’t use AWS tools, you must sign the request yourself. AWS CloudShell supports Signature Version 4, a protocol for authenticating inbound API requests. For more information about authenticating requests, see Signature Version 4 signing process in the AWS General Reference.

- **IAM role** – An IAM role is an IAM identity that you can create in your account that has specific permissions. An IAM role is similar to an IAM user in that it is an AWS identity with permissions policies that determine what the identity can and cannot do in AWS. However, instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it. Also, a role does not have standard long-term credentials such as a password or access keys associated with it. Instead, when you assume a role, it provides you with temporary security credentials for your role session. IAM roles with temporary credentials are useful in the following situations:

  - **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Creating a role for a third-party Identity Provider in the IAM User Guide. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see Permission sets in the AWS IAM Identity Center User Guide.

  - **AWS service access** – A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the IAM User Guide.

  - **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM role to grant permissions to applications running on Amazon EC2 instances in the IAM User Guide.

Managing AWS CloudShell access and usage with IAM policies

With the access management resources that can be provided by AWS Identity and Access Management (IAM), administrators can grant permissions to IAM users. That way, these users can access AWS CloudShell and use the environment’s features. Administrators can also create policies that specify at a granular level what actions those users can perform with the shell environment.

The quickest way for an administrator to grant access to users is through an AWS managed policy. An AWS managed policy is a standalone policy that's created and administered by AWS. The following AWS managed policy for AWS CloudShell can be attached to IAM identities:

- **AWSCloudShellFullAccess**: Grants permission to use AWS CloudShell with full access to all features.

The AWSCloudShellFullAccess policy uses the wildcard (*) character to give the IAM identity (user, role, or group) full access to CloudShell and features. You can also use the AWSCloudShellFullAccess policy as a template for custom policies that are more restrictive in terms of permitted user actions.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CloudShellUser",
            "Effect": "Allow",
            "Action": [
                "cloudshell:*"
            ]
        }
    ]
}
```
Managing allowable actions in AWS CloudShell using custom policies

To manage the actions that an IAM user can perform with CloudShell, create a custom policy that uses the AWSCloudShellFullAccess managed policy as a template. Alternatively, edit an inline policy that’s embedded in the relevant IAM identity (user, group, or role).

For example, you can allow users to access CloudShell but prevent them from uploading or downloading files within the shell environment. You can also explicitly deny users access to CloudShell.

**Important**
To launch AWS CloudShell from the AWS Management Console, an IAM user needs permissions for the following actions:

- CreateEnvironment
- CreateSession
- GetEnvironmentStatus
- StartEnvironment

If one of these actions isn't explicitly allowed by an attached policy, an IAM permissions error is returned when you try to launch CloudShell.

**AWS CloudShell permissions**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description of permission granted</th>
<th>Required to launch CloudShell?</th>
</tr>
</thead>
<tbody>
<tr>
<td>cloudshell:CreateEnvironment</td>
<td>Create a CloudShell environment, retrieve the layout at the start of the CloudShell session, and save the current layout from the webapp in the backend</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Managing AWS CloudShell access and usage with IAM policies

<table>
<thead>
<tr>
<th>Name</th>
<th>Description of permission granted</th>
<th>Required to launch CloudShell?</th>
</tr>
</thead>
<tbody>
<tr>
<td>cloudshell:CreateSession</td>
<td>Connect to a CloudShell environment from the AWS Management Console</td>
<td>Yes</td>
</tr>
<tr>
<td>cloudshell:GetEnvironmentStatus</td>
<td>Read the status of a CloudShell environment</td>
<td>Yes</td>
</tr>
<tr>
<td>cloudshell:DeleteEnvironment</td>
<td>Delete a CloudShell environment</td>
<td>No</td>
</tr>
<tr>
<td>cloudshell:GetFileDownloadUrls</td>
<td>Generate pre-signed Amazon S3 URLs which are used to download files through CloudShell using the CloudShell web interface</td>
<td>No</td>
</tr>
<tr>
<td>cloudshell:GetFileUploadUrls</td>
<td>Generate pre-signed Amazon S3 URLs which are used to upload files through CloudShell using the CloudShell web interface</td>
<td>No</td>
</tr>
<tr>
<td>cloudshell:PutCredentials</td>
<td>Forward the credentials used to log in to the AWS Management Console to CloudShell</td>
<td>No</td>
</tr>
<tr>
<td>cloudshell:StartEnvironment</td>
<td>Start a CloudShell environment that's stopped</td>
<td>Yes</td>
</tr>
<tr>
<td>cloudshell:StopEnvironment</td>
<td>Stop a CloudShell environment that's running</td>
<td>No</td>
</tr>
</tbody>
</table>

Examples of IAM policies for CloudShell

The following examples show how policies can be created to restrict who can access CloudShell. The examples also show the actions that can be performed in the shell environment.

This following policy enforces a complete denial of access to CloudShell and its features.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyCloudShell",
            "Effect": "Deny",
            "Action": [ "cloudshell:*" ],
            "Resource": "*"
        }
    ]
}
```

This following policy allows IAM users to access CloudShell but blocks them from generating pre-signed URLs for file upload and download. Users can still transfer files to and from the environment, using clients like `wget` for example.

This following policy enforces a complete denial of access to CloudShell and its features.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyCloudShell",
            "Effect": "Deny",
            "Action": [ "cloudshell:*" ],
            "Resource": "*"
        }
    ]
}
```

This following policy allows IAM users to access CloudShell but blocks them from generating pre-signed URLs for file upload and download. Users can still transfer files to and from the environment, using clients like `wget` for example.
The following policy allows IAM users to access CloudShell. However, the policy prevents the credentials that you used to log in to AWS Management Console from being forwarded to the CloudShell environment. IAM users with this policy need to manually configure their credentials within CloudShell.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CloudShellUser",
            "Effect": "Allow",
            "Action": [
                "cloudshell:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "DenyUploadDownload",
            "Effect": "Deny",
            "Action": [
                "cloudshell:GetFileDownloadUrls",
                "cloudshell:GetFileUploadUrls"
            ],
            "Resource": "*"
        }
    ]
}
```

Permissions for accessing AWS services

CloudShell uses the IAM credentials that you used to sign in to the AWS Management Console.

**Note**

To use the IAM credentials that you used to sign in to the AWS Management Console, you must have `cloudshell:PutCredentials` permission.

This pre-authentication feature of CloudShell makes it convenient to use AWS CLI. However, an IAM user still requires explicit permissions for the AWS services that are called from the command line.

For example, suppose that IAM users are required to create Amazon S3 buckets and upload files as objects to them. You can create a policy that explicitly allows those actions. The IAM console provides an interactive visual editor that guides through the process of building up a JSON-formatted policy document. After the policy is created, you can attach it to relevant IAM identity (user, group, or role).

For more information about attaching managed policies, see Adding IAM identity permissions (console) in the IAM User Guide.
Logging and monitoring in AWS CloudShell

This topic describes how you can log and monitor AWS CloudShell activity and performance with CloudTrail.

Monitoring activity with CloudTrail

AWS CloudShell is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or AWS service in AWS CloudShell. CloudTrail captures all API calls for AWS CloudShell as events. The calls captured include calls from the AWS CloudShell console and code calls to the AWS CloudShell API.

If you create a trail, you can enable the continuous delivery of CloudTrail events to an Amazon Simple Storage Service (Amazon S3) bucket. This includes events for AWS CloudShell.

If you don't configure a trail, you can still view the most recent events in the CloudTrail console in Event history. Using the information collected by CloudTrail, you can discover a variety of information about a request. For example, you can determine the request that was made to AWS CloudShell, you can learn the IP address that the request was made from, who made the request, and when it was made.

AWS CloudShell in CloudTrail

AWS CloudShell supports logging the following actions as events in CloudTrail log files:

- createEnvironment
- createSession
- deleteEnvironment
- deleteSession
- getEnvironmentStatus*
- getFileDownloadUrls*
- getFileUploadUrls*
- getLayout*
- putCredentials
- redeemCode
- sendHeartBeat
- startEnvironment
- stopEnvironment
- updateLayout

*Non-mutating (read-only) API calls.

Events that include the word "Environment" in their names relate to the lifecycle of the compute environment that hosts the shell experience.

Events that include the word "Layout" restore all the browser tabs in the CloudShell terminal.

The sendHeartBeat event occurs to confirm that the session is active. The putCredentials event occurs when the credentials the user signed in to console with are forwarded to AWS CloudShell.

The deleteSession event deletes the session in one CloudShell tab that's running in the current browser tab. The updateLayout event saves the current layout from the webapp in the backend. The getLayout event retrieves the layout at the start of the session.
EventBridge rules for AWS CloudShell actions

With EventBridge rules, you specify a target action to take when EventBridge receives an event that matches the rule. You can define a rule that specifies a target action to take based on an AWS CloudShell action that's recorded as an event in a CloudTrail log file.

For example, you can create EventBridge rules with AWS CLI using the put-rule command. A put-rule call must contain at least an EventPattern or ScheduleExpression. Rules with EventPatterns are triggered when a matching event is observed. The EventPattern for AWS CloudShell events:

```
{ "source": [ "aws.cloudshell" ], "detail-type": [ "AWS API Call via CloudTrail" ],
"detail": { "eventSource": [ "cloudshell.amazonaws.com" ] }
}
```

For more information, see Events and Event Patterns in EventBridge in the Amazon EventBridge User Guide.

Compliance validation for AWS CloudShell

Third-party auditors assess the security and compliance of AWS services as part of multiple AWS compliance programs.

AWS CloudShell is in scope with the following compliance programs:

**SOC**

AWS System and Organization Controls (SOC) Reports are independent third-party examination reports that demonstrate how AWS achieves key compliance controls and objectives.

<table>
<thead>
<tr>
<th>Service</th>
<th>SDK</th>
<th>SOC 1,2,3</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS CloudShell</td>
<td>CloudShell</td>
<td>✓</td>
</tr>
</tbody>
</table>

**PCI**

The Payment Card Industry Data Security Standard (PCI DSS) is a proprietary information security standard administered by the PCI Security Standards Council, which was founded by American Express, Discover Financial Services, JCB International, MasterCard Worldwide and Visa Inc.

<table>
<thead>
<tr>
<th>Service</th>
<th>SDK</th>
<th>PCI</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS CloudShell</td>
<td>CloudShell</td>
<td>✓</td>
</tr>
</tbody>
</table>

**ISO and CSA STAR Certifications and Services**


<table>
<thead>
<tr>
<th>Service</th>
<th>SDK</th>
<th>ISO and CSA STAR Certifications and Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS CloudShell</td>
<td>CloudShell</td>
<td>✓</td>
</tr>
</tbody>
</table>
FedRamp

The Federal Risk and Authorization Management Program (FedRAMP) is a US government-wide program that delivers a standard approach to the security assessment, authorization, and continuous monitoring for cloud products and services.

<table>
<thead>
<tr>
<th>Service</th>
<th>SDK</th>
<th>FedRAMP Moderate (East/West)</th>
<th>FedRAMP High (GovCloud)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS CloudShell</td>
<td>CloudShell</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

DoD CC SRG

The Department of Defense (DoD) Cloud Computing Security Requirements Guide (SRG) provides a standardized assessment and authorization process for cloud service providers (CSPs) to gain a DoD provisional authorization, so that they can serve DoD customers.

Services going through DoD CC SRG assessment and authorization will have the following status:

- **Third-Party Assessment Organization (3PAO) Assessment:** This service is currently undergoing an assessment by our third-party assessor.
- **Joint Authorization Board (JAB) Review:** This service is currently undergoing a JAB review.
- **Defense Information Systems Agency (DISA) Review:** This service is currently undergoing a DISA review.

<table>
<thead>
<tr>
<th>Service</th>
<th>SDK</th>
<th>DoD CC SRG IL2 (East/West)</th>
<th>DoD CC SRG IL2 (GovCloud)</th>
<th>DoD CC SRG IL4 (GovCloud)</th>
<th>DoD CC SRG IL5 (GovCloud)</th>
<th>DoD CC SRG IL6 (AWS Secret Region)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS CloudShell</td>
<td>CloudShell</td>
<td>3PAO Assessment</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

HIPAA BAA

The Health Insurance Portability and Accountability Act of 1996 (HIPAA) is a federal law that required the creation of national standards to protect sensitive patient health information from being disclosed without the patient's consent or knowledge.

AWS enables covered entities and their business associates subject to HIPAA to securely process, store, and transmit protected health information (PHI). Additionally, as of July 2013, AWS offers a standardized Business Associate Addendum (BAA) for such customers.

<table>
<thead>
<tr>
<th>Service</th>
<th>SDK</th>
<th>HIPAA BAA</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS CloudShell</td>
<td>CloudShell</td>
<td>✓</td>
</tr>
</tbody>
</table>

IRAP

The Information Security Registered Assessors Program (IRAP) enables Australian Government customers to validate that appropriate controls are in place and determine the appropriate responsibility model for

<table>
<thead>
<tr>
<th>Service</th>
<th>SDK</th>
<th>HIPAA BAA</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS CloudShell</td>
<td>CloudShell</td>
<td>✓</td>
</tr>
</tbody>
</table>
addressing the requirements of the Australian Government Information Security Manual (ISM) produced by the Australian Cyber Security Centre (ACSC).

<table>
<thead>
<tr>
<th>Service</th>
<th>Namespace*</th>
<th>IRAP protected</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS CloudShell</td>
<td>N/A</td>
<td>✓</td>
</tr>
</tbody>
</table>

*Namespaces help you identify services across your AWS environment. For example, when you create IAM policies, work with Amazon Resource Names (ARNs), and read AWS CloudTrail logs.

**MTCS**

The Multi-Tier Cloud Security (MTCS) is an operational Singapore security management Standard (SPRING SS 584), based on ISO 27001/02 Information Security Management System (ISMS) standards.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS CloudShell</td>
<td>CloudShell</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**C5**

Cloud Computing Compliance Controls Catalog (C5) is a German Government-backed attestation scheme introduced in Germany by the Federal Office for Information Security (BSI) to help organizations demonstrate operational security against common cyber-attacks when using cloud services within the context of the German Government's "Security Recommendations for Cloud Providers".

<table>
<thead>
<tr>
<th>Service</th>
<th>SDK</th>
<th>C5</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS CloudShell</td>
<td>CloudShell</td>
<td>✓</td>
</tr>
</tbody>
</table>

**ENS High**

The ENS (Esquema Nacional de Seguridad) accreditation scheme has been developed by the Ministry of Finance and Public Administration and the CCN (National Cryptologic Centre). This comprises of basic principles and minimum requirements necessary for the adequate protection of information.

<table>
<thead>
<tr>
<th>Service</th>
<th>SDK</th>
<th>ENS High</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS CloudShell</td>
<td>CloudShell</td>
<td>✓</td>
</tr>
</tbody>
</table>

**FINMA**

The Swiss Financial Market Supervisory Authority (FINMA) is Switzerland’s independent financial-markets regulator. AWS's alignment with FINMA requirements demonstrates our continuous commitment to meeting the heightened expectations for cloud service providers set by Swiss financial services regulators and customers.
PiTuKri

AWS alignment with PiTuKri requirements demonstrates our continuous commitment to meeting the heightened expectations for cloud service providers set by Finnish Transport and Communications Agency, Traficom.

<table>
<thead>
<tr>
<th>Service</th>
<th>SDK</th>
<th>PiTuKri</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS CloudShell</td>
<td>CloudShell</td>
<td>✓</td>
</tr>
</tbody>
</table>

For a list of AWS services that are in scope of specific compliance programs, see [AWS Services in Scope by Compliance Program](#). For general information, see [AWS Compliance Programs](#).

You can download third-party audit reports by using AWS Artifact. For more information, see [Downloading Reports in AWS Artifact](#).

Your compliance responsibility when using AWS CloudShell is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- **Security and Compliance Quick Start Guides** – These deployment guides discuss architectural considerations and provide steps for deploying security-focused and compliance-focused baseline environments on AWS.
- **Architecting for HIPAA Security and Compliance Whitepaper** – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- **AWS Compliance Resources** – This collection of workbooks and guides might apply to your industry and location.
- **Evaluating Resources with Rules** in the [AWS Config Developer Guide](#) – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- **AWS Security Hub** – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

Resilience in AWS CloudShell

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see [AWS Global Infrastructure](#).

In addition to the AWS global infrastructure, AWS CloudShell supports specific features to support your data resiliency and backup needs.
Commit files you create and add to AWS CodeCommit. This is a version control service hosted by Amazon Web Services that you can use to privately store and manage assets in the cloud. These assets can consist of documents, source code, and binary files. For more information, see Tutorial: Using CodeCommit in AWS CloudShell (p. 26).

Use AWS CLI calls to specify files in your home directory in AWS CloudShell and add them as objects in Amazon S3 buckets. For an example, see the getting started tutorial (p. 11).

Infrastructure security in AWS CloudShell

As a managed service, AWS CloudShell is protected by AWS global network security. For information about AWS security services and how AWS protects infrastructure, see AWS Cloud Security. To design your AWS environment using the best practices for infrastructure security, see Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

You use AWS published API calls to access AWS CloudShell through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

Note
By default, AWS CloudShell automatically install security patches for the system packages of your compute environments.

Configuration and vulnerability analysis in AWS CloudShell

It's the responsibility of the AWS CloudShell user to ensure that any software they installed in the compute environment is patched and up to date.

Security best practices for AWS CloudShell

The following best practices are general guidelines and don't represent a complete security solution. Because these best practices might not be appropriate or sufficient for your environment, treat them as helpful considerations instead of prescriptions.

Some security best practices for AWS CloudShell

- Use IAM permissions and policies to control access to AWS CloudShell and ensure users can perform only those actions (downloading and uploading files, for example) required by their role. For more information, see Managing AWS CloudShell access and usage with IAM policies (p. 43).
- Don't include sensitive data in your IAM entities such as users, roles, or session names.
• Keep Safe Paste feature enabled to catch potential security risks in text you've copied from external sources. Safe Paste is enabled by default. For more information, see Using Safe Paste for multiline text (p. 39).
• Be familiar with the Shared Security Responsibility Model if you installed third-party applications to the compute environment of AWS CloudShell.
• Prepare rollback mechanisms before editing shell scripts that affect the user's shell experience. For more information, see Modifying your shell with scripts (p. 64).
• Store your code securely in a version control system, for example, AWS CodeCommit.

AWS CloudShell Security FAQs

Answers to frequently asked questions about security for this AWS service.

• What are the AWS processes and technologies that are used when you launch CloudShell and start a shell session? (p. 53)
• Is it possible to restrict network access to CloudShell? (p. 53)
• Can I customize my CloudShell environment? (p. 53)
• Where is my $HOME directory actually stored in the AWS Cloud? (p. 54)
• Is it possible to encrypt my $HOME directory? (p. 54)
• Can I run a virus scan on my $HOME directory? (p. 54)

What are the AWS processes and technologies that are used when you launch CloudShell and start a shell session?

When signing into AWS Management Console, you enter your IAM user credentials. And, when you launch CloudShell from the console interface, these credentials are used in calls to the CloudShell API that create a compute environment for the service. An AWS Systems Manager session is then created for the compute environment, and CloudShell sends commands to that session.

Is it possible to restrict network access to CloudShell?

It might be possible to restrict network access to CloudShell by connecting with your network provider. Alternatively, you can use IAM permissions, explicitly deny access to CloudShell or not provide any IAM permissions, and use implicit deny IAM feature. For more information, see Managing AWS CloudShell access and usage with IAM policies (p. 43).

Can I customize my CloudShell environment?

You can download and install utilities and other third-party software for your CloudShell environment. Only software that's installed in your $HOME directory is persisted between sessions.

As defined by the AWS shared responsibility model, you are responsible for the necessary configuration and management of applications that you install.

Back to list of security FAQs (p. 53)
Where is my $HOME directory actually stored in the AWS Cloud?

The infrastructure for storing data in your $HOME is provided by Amazon S3.

Back to list of security FAQs (p. 53)

Is it possible to encrypt my $HOME directory?

Your data in your $HOME directory is already encrypted using Amazon S3 Encryption.

Back to list of security FAQs (p. 53)

Can I run a virus scan on my $HOME directory?

At present, it's not possible to run a virus scan of your $HOME directory. Support for this feature is under review.

Back to list of security FAQs (p. 53)
AWS CloudShell compute environment: specifications and software

When you launch AWS CloudShell, a compute environment that's based on Amazon Linux 2 is created to host the shell experience. The environment is configured with compute resources (vCPU and memory) (p. 55) and provides a wide range of pre-installed software (p. 56) that can be accessed from the command line interface. You can also configure your default environment by installing software and modifying shell scripts.

Compute environment resources

Each AWS CloudShell compute environment is assigned the following CPU and memory resources:

- 1 vCPU (virtual central processing unit)
- 2-GiB RAM

And, the environment is provisioned with the following storage configuration:

- 1-GB persistent storage (storage persists after the session ends)

For more information, see Persistent storage (p. 74).

Important
Currently, the AWS CloudShell compute environment doesn't support Docker containers.

CloudShell network requirements

WebSockets

CloudShell depends on the WebSocket protocol, which allows two-way interactive communication between the user's web browser and the CloudShell service in the AWS Cloud. If you're using a browser in a private network, secure access to the internet is probably facilitated by proxy servers and firewalls. WebSocket communication can usually traverse proxy servers without a problem. But in some cases, proxy servers prevent WebSockets from working correctly. If this issue occurs, your CloudShell interface reports the following error: Failed to open sessions : Timed out while opening the session.

If this error occurs repeatedly, see the documentation for your proxy server to ensure that it's configured to allow WebSockets. Alternatively, you can contact your network's system administrator.

Note
If you want to define granular permissions by allow-listing specific URLs, you can add part of the URL that the AWS Systems Manager session uses to open a WebSocket connection for sending input and receiving outputs. (Your AWS CloudShell commands are sent to that Systems Manager session.)
The format for this StreamUrl used by Systems Manager is `wss://ssmmessages.region.amazonaws.com/v1/data-channel/session-id?stream=(input|output)`.

The `region` represents the Region identifier for an AWS Region supported by AWS Systems Manager, such as `us-east-2` for the US East (Ohio) Region. Because the `session-id` is created after a particular Systems Manager session is successfully started, you can only specify `wss://ssmmessages.region.amazonaws.com` when updating your URL allowlist. For more information, see the `StartSession` operation in the AWS Systems Manager API Reference.

---

### Pre-installed software

**Note**

Because the AWS CloudShell development environment is regularly updated to provide access to the latest software, we don't provide specific version numbers in this documentation. Instead, we describe how you can check which version is installed. To check the installed version, enter the program name followed by the `--version` option (for example, `git --version`).

### Shells

#### Pre-installed shells

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Version information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bash</td>
<td>The Bash shell is the default shell application for AWS CloudShell.</td>
<td><code>bash --version</code></td>
</tr>
<tr>
<td>PowerShell (pwsh)</td>
<td>Offering a command line interface and scripting language support, PowerShell is built on top of Microsoft's .NET Command Language Runtime. PowerShell uses lightweight commands called cmdlets that accept and return .NET objects.</td>
<td><code>pwsh --version</code></td>
</tr>
<tr>
<td>Z Shell (zsh)</td>
<td>The Z Shell, also known as zsh, is an extended version of the Bourne Shell that offers enhanced customization support for themes and plugins.</td>
<td><code>zsh --version</code></td>
</tr>
</tbody>
</table>

### AWS command line interfaces (CLI)

#### CLI

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Version information</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS CDK Toolkit CLI</td>
<td>The AWS CDK Toolkit, the CLI command, <code>cdk</code>, is the primary tool that interacts with your AWS CDK app. It executes your app, interrogates the</td>
<td><code>cdk --version</code></td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Version information</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>AWS CLI</td>
<td>The AWS CLI is a command line interface that you can use to manage multiple AWS services from the command line and automate them using scripts. For more information, see <strong>Working with AWS services in AWS CloudShell</strong> (p. 35). For information about how you can ensure that you're using the most up-to-date version AWS CLI version 2, see <strong>Installing AWS CLI to your home directory</strong> (p. 63).</td>
<td>aws --version</td>
</tr>
<tr>
<td>EB CLI</td>
<td>The AWS Elastic Beanstalk CLI provides a command line interface to simplify creating, updating, and monitoring environments from a local repository. For more information, see <strong>Using the Elastic Beanstalk command line interface (EB CLI)</strong> in the <strong>AWS Elastic Beanstalk Developer Guide</strong>.</td>
<td>eb --version</td>
</tr>
<tr>
<td>Amazon ECS CLI</td>
<td>Amazon Elastic Container Service (Amazon ECS) command line interface (CLI) provides high-level commands to simplify creating, updating, and monitoring clusters and tasks. For more information, see <strong>Using the Amazon ECS Command Line Interface</strong> in the <strong>Amazon Elastic Container Service Developer Guide</strong>.</td>
<td>ecs-cli --version</td>
</tr>
</tbody>
</table>
### Runtimes and AWS SDKs: Node.js and Python 3

#### Runtimes and AWS SDKs

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Version information</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS SAM CLI</td>
<td>AWS SAM CLI is a command line tool that operates on an AWS Serverless Application Model template and application code. You can perform several tasks. These include invoking Lambda functions locally, creating a deployment package for your serverless application, and deploying your serverless application to the AWS Cloud. For more information, see the <a href="https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/sam-reference.html">AWS SAM CLI command reference</a> in the <a href="https">AWS Serverless Application Model Developer Guide</a>.</td>
<td><code>sam --version</code></td>
</tr>
</tbody>
</table>

#### Node.js (with npm)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Version information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node.js (with npm)</td>
<td>Node.js is a JavaScript runtime that's designed to make it easier to apply asynchronous programming techniques. For more information, see the <a href="https://nodejs.org">documentation on the official Node.js site</a>. npm is a package manager that provides access to an online registry of JavaScript modules. For more information, see the documentation.</td>
<td>• Node.js: node --version</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• npm: npm --version</td>
</tr>
</tbody>
</table>
## Development tools and shell utilities

### Name | Description | Version information
--- | --- | ---
| SDK for JavaScript in Node.js | The software development kit (SDK) helps simplify coding by providing JavaScript objects for AWS services including Amazon S3, Amazon EC2, DynamoDB, and Amazon SWF. For more information, see the [AWS SDK for JavaScript Developer Guide](https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/). | npm -g ls --depth 0 2>/dev/null | grep aws-sdk |
| Python | Python 3 and Python 2 are both ready to use in the shell environment. Python 3 is now considered the default version of the programming language (support for Python 2 ended in January 2020). For more information, see the [documentation on the official Python site](https://docs.python.org/). Also, pre-installed is pip, the package installer for Python. You can use this command line program to install Python packages from the online indexes such as the Python Package Index. For more information, see the [documentation provided by the Python Packaging Authority](https://pypi.org/). | • Python 2: python --version  
• Python 3: python3 --version  
• pip: pip3 --version |
| SDK for Python (Boto3) | Boto is the software development kit (SDK) that Python developers use to create, configure, and manage AWS services, such as Amazon EC2 and Amazon S3. The SDK provides an easy-to-use, object-oriented API, as well as low-level access to AWS services. For more information, see the [Boto3 documentation](https://boto3.amazonaws.com/v1/documentation/api/latest/index.html). | pip3 list | grep boto3 |

### Development tools and shell utilities

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Version information</th>
</tr>
</thead>
<tbody>
<tr>
<td>bash-completion</td>
<td>bash-completion is a collection of shell functions that allow</td>
<td>yum info bash-completion</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Version information</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>the autocompletion of partially typed commands or arguments by pressing the <strong>Tab</strong> key. You can find the packages that bash-completion supports in <code>/usr/share/bash-completion/completions</code>.</td>
<td>pip3 list</td>
</tr>
<tr>
<td></td>
<td>To set up autocomplete for a package's commands, the program file must be sourced. For example, to set up autocomplete for Git commands, add the following line to <code>.bashrc</code> so the feature is available whenever your AWS CloudShell session starts:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>source /usr/share/bash-completion/completions/git</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>If you want to use custom completion scripts, add them to your persistent home directory (<code>$HOME</code>) and source them directly in <code>.bashrc</code>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For more information, see the project's <strong>README</strong> page on GitHub.</td>
<td></td>
</tr>
<tr>
<td>CodeCommit utility for Git</td>
<td><code>git-remote-codecommit</code> is a utility that provides a simple method for pushing and pulling code from CodeCommit repositories by extending Git. It's the recommended method for supporting connections that are made with federated access, identity providers, and temporary credentials.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For more information, see <a href="https://docs.aws.amazon.com/codecommit/latest/userguide/sample-configuration.html#sample-config-hybrid-1">Setup steps for HTTPS connections to AWS CodeCommit with git-remote-codecommit</a> in the AWS CodeCommit User Guide.</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Version information</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Git</td>
<td>Git is a distributed version control system that supports modern software development practices through branch workflows and content staging. For more information, see the <a href="#">documentation page on Git's official site.</a></td>
<td><code>git --version</code></td>
</tr>
<tr>
<td>iputils</td>
<td>The iputils package contains utilities for Linux networking. For more information about the utilities provided, see the <a href="#">iputils repository on GitHub.</a></td>
<td>Examples for an iputils tool: <code>arping -V</code></td>
</tr>
<tr>
<td>jq</td>
<td>The jq utility parses JSON-formatted data to produce output that's modified by command line filters. For more information, see the <a href="#">jq manual hosted on GitHub.</a></td>
<td><code>jq --version</code></td>
</tr>
<tr>
<td>kubectl</td>
<td>kubectl is a command line tool for communicating with a Kubernetes cluster's control plane, using the Kubernetes API.</td>
<td><code>kubectl --version</code></td>
</tr>
<tr>
<td>make</td>
<td>The make utility uses makefiles to automate sets of tasks and organize code compilation. For more information, see the <a href="#">GNU Make documentation.</a></td>
<td><code>make --version</code></td>
</tr>
<tr>
<td>man</td>
<td>The man command provides manual pages for command line utilities and tools. For example, <code>man ls</code> returns the manual page for the <code>ls</code> command that lists the contents of directories. For more information, see the <a href="#">Wikipedia entry on man page.</a></td>
<td><code>man --version</code></td>
</tr>
<tr>
<td>nano</td>
<td>nano is a small and user-friendly editor for text-based interface. For more information, see the <a href="#">GNU nano documentation.</a></td>
<td><code>nano --version</code></td>
</tr>
<tr>
<td>procps</td>
<td>procps is a system administration utility that you can use to monitor and halt currently running processes. For more information, see the <a href="#">README file that lists programs that can be run with procps.</a></td>
<td><code>ps --version</code></td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Version information</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>SSH client</td>
<td>SSH clients use the secure shell protocol for encrypted communications with a remote computer. OpenSSH is the SSH client that's pre-installed. For more information, see the <a href="https://www.openssh.org">OpenSSH site</a> maintained by the OpenBSD.</td>
<td><code>ssh -V</code></td>
</tr>
<tr>
<td>sudo</td>
<td>With the sudo utility, users can run a program with the security permissions of another user, typically the superuser. Sudo is useful when you need to install applications as a system administrator. For more information, see the <a href="https://man7.org/linux/man-pages/man1/sudo.1.html">Sudo Manual</a>.</td>
<td><code>sudo --version</code></td>
</tr>
<tr>
<td>tar</td>
<td>tar is a command line utility that you can use to group multiple files in a single archive file (often called a tarball). For more information, see the <a href="https://www.gnu.org/software/tar/manual/tar.html">GNU tar documentation</a>.</td>
<td><code>tar --version</code></td>
</tr>
<tr>
<td>tmux</td>
<td>tmux is a terminal multiplexer that you can use to run different programs simultaneous in multiple windows. For more information, see <a href="https://www.thegeekstuff.com/2019/05/tmux-commands-and-tips/">a blog that provides a concise introduction to tmux</a>.</td>
<td><code>tmux -V</code></td>
</tr>
<tr>
<td>unzip</td>
<td>For more information, see <a href="https://www.gnu.org/software/zlib/zlib1.html">zip/unzip (p. 59)</a>.</td>
<td></td>
</tr>
<tr>
<td>vim</td>
<td>vim is a customizable editor that you can interact with through a text-based interface. For more information, see the <a href="https://vim.org">documentation resources provided on vim.org</a>.</td>
<td><code>vim --version</code></td>
</tr>
<tr>
<td>wget</td>
<td>wget is a computer program used to retrieve content from web servers specified by endpoints in the command line. For more information, see the <a href="http://www.gnu.org/software/wget/manual/wget.html">GNU Wget documentation</a>.</td>
<td><code>wget --version</code></td>
</tr>
</tbody>
</table>
## Installing AWS CLI to your home directory

Like the rest of the software that's pre-installed in your CloudShell environment, the AWS CLI tool is updated automatically with scheduled upgrades and security patches. If you want to ensure that you have the most up-to-date version of AWS CLI, you can choose to manually install the tool in the shell's home directory.

**Important**
You need to manually install your copy of AWS CLI in the home directory so that it's available the next time you start a CloudShell session. This installation is needed because files that are added to directories outside of $HOME are deleted after you finish a shell session. Also, after you install this copy of AWS CLI, it isn't automatically updated. In other words, it's your responsibility to manage updates and security patches. For more information about the AWS Shared Responsibility Model, see [Data protection in AWS CloudShell](p. 40).

**To install AWS CLI**

1. In the CloudShell command line, use the `curl` command to transfer a zipped copy of the AWS CLI installed to the shell:

   ```bash
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
   ```

2. Unzip the zipped folder:

   ```bash
   unzip awscliv2.zip
   ```

3. To add the tool to a specified folder, run the AWS CLI installer:

   ```bash
   sudo ./aws/install --install-dir /home/cloudshell-user/usr/local/aws-cli --bin-dir /home/cloudshell-user/usr/local/bin
   ```

   If it's installed successfully, the command line displays the following message:

   ```bash
   You can now run: /home/cloudshell-user/usr/local/bin/aws --version
   ```

4. For your own convenience, we recommend that you also update the `PATH` environmental variable so that you don't need to specify the path to your installation of the tool when running `aws` commands:

   ```bash
   export PATH=/home/cloudshell-user/usr/local/bin:$PATH
   ```
Note
If you undo this change to PATH, aws commands that don't feature a specified path use the pre-installed version of AWS CLI by default.

Installing third-party software on your shell environment

Note
We recommend that you review the Shared Security Responsibility Model before you install any third-party applications to the AWS CloudShell's compute environment.

By default, all AWS CloudShell users have sudo permissions. Therefore, you can use the sudo command to install software that's not already available in the shell's compute environment. For example, you can use sudo with the YUM package-management utility to install cowsay, which generates ASCII art pictures of a cow with a message:

```bash
sudo yum install cowsay
```

You can then launch the newly installed program by typing `echo "Welcome to AWS CloudShell" | cowsay`.

Important
Package manage utilities such as yum install programs in directories (/usr/bin, for example), which are recycled when your shell session ends. This means additional software is installed and used on a per-session basis.

Modifying your shell with scripts

If you want to modify the default shell environment, you can edit a shell script that runs every time the shell environment starts up. The .bashrc script runs whenever the default bash shell starts up.

Warning
If you incorrectly modify your .bashrc file, you might not be able to access your shell environment afterward. It's good practice to make a copy of the file before editing. You can also mitigate risk by opening two shells when editing .bashrc. If you lose access in one shell, you're still logged in into the other shell and can roll back any changes. If you do lose access after incorrectly modifying .bashrc or any other file, you can return AWS CloudShell to its default settings by deleting your home directory (p. 19).

In the procedure, you'll modify the .bashrc script so that your shell environment switches automatically to running the Z shell.

1. Open the .bashrc using a text editor (Vim, for example):

   ```
   vim .bashrc
   ```

2. In the editor interface, press the `I` key to start editing, and then add the following:

   ```
   zsh
   ```

3. To exit and save the edited .bashrc file, press `Esc` to enter the Vim command mode and enter the following:
4. Use the `source` command to reload the `.bashrc` file:

```bash
source .bashrc
```

When the command line interface becomes available again, the prompt symbol has changed to `%` to indicate that you're now using the Z shell.

## AWS CloudShell migrating from Amazon Linux 2 to Amazon Linux 2023


After CloudShell migrates from AL2 to AL2023, you can continue to access your existing CloudShell environment with all tools provided by CloudShell. For more information about available tools, see [Pre-installed software](#) (p. 56). If you've modified content of your home folder, we recommend that you create a backup if you lose access to it after the migration. For more information about how to create a backup of a home directory, see [Create a home directory backup](#) (p. 17).

AL2023 provides several improvements to development tools, including newer versions of packages such as Node.js 18 and Python 3.9.

**Note**

In AL2023, Python 2 will no longer be shipped with your CloudShell environment.


If you've any questions, contact [AWS Support](https://aws.amazon.com/support/). You can also search for answers and post questions in [AWS re:Post](https://aws.amazon.com/repost/). When you enter AWS re:Post, you might be required to sign in to AWS.

## AWS CloudShell Migration FAQs

The following are answers to some common questions about the migration from AL2 to AL2023 with AWS CloudShell.

- Will this migration affect any of my other AWS resources, like Amazon EC2 instances running on AL2? (p. 65)
- What are the packages that will be changed with the migration to AL2023? (p. 66)
- Can I opt-out from migration? (p. 66)
- Can I create a backup of my AWS CloudShell environment? (p. 66)

**Will this migration affect any of my other AWS resources, like Amazon EC2 instances running on AL2?**

No service or resource other than your AWS CloudShell environment is affected by this migration. This includes resources that you may have created or accessed from within AWS CloudShell. For example, if you have created an Amazon EC2 instance running on AL2 this will not be migrated to AL2023.
What are the packages that will be changed with the migration to AL2023?

AWS CloudShell environments currently include pre-installed software. To learn about the complete list of pre-installed software, see Pre-installed software. After the migration, AWS CloudShell will continue delivering these packages, with an exception of Python 2. For the complete difference between the packages provided by AL2 and AL2023, see Comparing AL2 and AL2023. For customers with specific package and version requirements that will no longer be met after the migration to AL2023, we recommend reaching out to AWS Support to submit a request.

Can I opt-out from migration

The answer is NO. AWS CloudShell environments are managed by AWS, therefore, all the environments will be upgraded to AL2023.

Can I create a backup of my AWS CloudShell environment?

AWS CloudShell will continue to persist the user home directory. For more information, see Service quotas and restrictions for AWS CloudShell. If you have any files or configurations stored in your home folder and if you want to create a backup for the same, complete Step 6: Create a home directory backup.
Troubleshooting AWS CloudShell

While using AWS CloudShell, you might encounter issues, such as when you launch CloudShell or perform key tasks using the shell command line interface. The information that's covered in this chapter covers how to troubleshoot some of the common issues that you might encounter.

For answers to a variety of questions about CloudShell, see the AWS CloudShell FAQs. You can also search for answers and post questions in the AWS CloudShell Discussion Forum. When you enter this forum, you might be required to sign in to AWS. You can also contact us directly.

Troubleshooting errors

When you come across any of the following indexed errors, you can use the following solutions to resolve these errors.

Topics

- Unable to start the environment. To retry, refresh the browser or restart by selecting Actions, Restart AWS CloudShell (p. 67)
- Unable to start the environment. You don't have required permissions. Ask your IAM administrator to grant access to AWS CloudShell (p. 67)
- Unable to access AWS CloudShell command line (p. 68)
- Unable to ping external IP addresses (p. 68)
- There were some issues preparing your terminal. (p. 68)
- Arrow keys not working correctly in PowerShell (p. 68)
- Unsupported Web Sockets cause a failure to start CloudShell sessions (p. 69)
- Unable to import the AWSPowerShell.NetCore module (p. 70)

Unable to start the environment. To retry, refresh the browser or restart by selecting Actions, Restart AWS CloudShell

Issue: When you attempt to launch AWS CloudShell from the AWS Management Console, you're denied access even after you have required permissions from your IAM administrator and you have refreshed your browser or restarted CloudShell.

Solution: Contact AWS Support.

Unable to start the environment. You don't have required permissions. Ask your IAM administrator to grant access to AWS CloudShell

Issue: When you attempt to launch AWS CloudShell from the AWS Management Console, you're denied access and notified that you don't have required permissions.
Unable to access AWS CloudShell command line

**Cause:** The IAM identity that you're using to access AWS CloudShell lacks the necessary IAM permissions.

**Solution:** Request your IAM administrator to provide you with the necessary permissions. They can do this either through adding an attached AWS managed policy (AWSCloudShellFullAccess) or an embedded inline policy. For more information, see [Managing AWS CloudShell access and usage with IAM policies](#).

Unable to ping external IP addresses

**Issue:** When you run a ping command from the command line (for example, `ping amazon.com`), you receive the following message.

```
ping: socket: Operation not permitted
```

**Cause:** The ping utility uses Internet Control Message Protocol (ICMP) to send echo requests packets to a target host. It waits for an echo to reply from the target. Because the ICMP protocol isn't enabled in AWS CloudShell, the ping utility doesn't operate in the shell's compute environment.

There were some issues preparing your terminal

**Issue:** When trying to access AWS CloudShell using the Microsoft Edge browser, you can't start a shell session, and the browser displays an error message.

**Cause:** AWS CloudShell isn't compatible with earlier versions of Microsoft Edge. You can access AWS CloudShell using the latest four major versions of [supported browsers](#).

**Solution:** Install an updated version of Edge browser from the [Microsoft site](#).

Arrow keys not working correctly in PowerShell

**Issue:** In normal operation, you can use arrow keys to navigate the command line interface and scan backwards and forwards through your command history. But, when you press arrow keys in certain versions of PowerShell on AWS CloudShell, letters might be incorrectly outputted.

**Cause:** The situation where arrow keys incorrectly output letters is a known issue with PowerShell 7.2.x versions running on Linux.

**Solution:** To strip out escape sequences that modify the behavior of arrow keys, edit the PowerShell profile file and set the `$PSStyle` variable to `PlainText`.
1. In the AWS CloudShell command line, enter the following command to open the profile file.

   `vim ~/.config/powershell/Microsoft.PowerShell_profile.ps1`

   **Note**
   If you're already in PowerShell, you can also open the profile file in the editor with the following command.

   `vim $PROFILE`

2. In the editor, go to the end of the file's existing text, press `i` to enter **Insert** mode, and then add the following statement.

   `$PSStyle.OutputRendering = 'PlainText'`

3. After you make the edit, press `Esc` to enter the command mode. Next, enter the following command to save the file and exit the editor.

   `:wq`

   **Note**
   Your changes take effect the next time you start PowerShell.

   *(back to top (p. 67))*

### Unsupported Web Sockets cause a failure to start CloudShell sessions

**Issue:** When trying to start AWS CloudShell, you repeatedly receive the following message: Failed to open sessions : Timed out while opening the session.

**Cause:** CloudShell depends on the WebSocket protocol, which allows for two-way interactive communication between your web browser and AWS CloudShell. If you're using a browser in a private network, secure access to the internet is probably facilitated by proxy servers and firewalls. WebSocket communication can usually traverse proxy servers without a problem. But, in some cases, proxy servers prevent WebSockets from working correctly. If this issue occurs, CloudShell can't start a shell session and the attempt to connect eventually times out.

**Solution:** A connection timeout might be caused by an issue other than unsupported WebSockets. If this is the case, first refresh the browser window where the CloudShell command line interface is located.

If you're still getting timeout errors after the refresh, see the documentation for your proxy server. And, make sure that your proxy server is configured to allow Web Sockets. Alternatively, contact your network’s system administrator.

**Note**
Say that you want to define granular permissions by allowing listing specific URLs. You can add part of the URL that the AWS Systems Manager session uses to open a WebSocket connection for sending input and receiving outputs. Your AWS CloudShell commands are sent to that Systems Manager session.

The format for this StreamUrl that's used by Systems Manager is `wss://ssmmessages.region.amazonaws.com/v1/data-channel/session-id?stream=(input|output)`.

The **region** represents the Region identifier for an AWS Region that's supported by AWS Systems Manager. For example, us-east-2 is the Region identifier for the US East (Ohio) Region.
Unable to import the AWSPowerShell.NetCore module

Issue: When you import the AWSPowerShell.NetCore module in PowerShell by `Import-Module -Name AWSPowerShell.NetCore`, you receive the following error message:

```
Import-Module: The specified module 'AWSPowerShell.NetCore' was not loaded because no valid module file was found in any module directory.
```

Cause: The AWSPowerShell.NetCore module is replaced by the per-service AWS.Tools modules in AWS CloudShell.

Solution: Any explicit import statements might no longer be required or need to be changed to the related per-service AWS.Tools module.

Example

Example

- For most cases, as long as no .Net types are used, you don’t need any explicit import statement. The following are examples of import statements.
  - `Get-S3Bucket`
  - `(Get-EC2Instance).Instances`

- If .Net types are used, import the service-level module (AWS.Tools.<Service>). The following is example syntax.

```powershell
Import-Module -Name AWS.Tools.EC2
$instanceTag = [Amazon.EC2.Model.Tag]::new("Environment","Dev")
```

```powershell
Import-Module -Name AWS.Tools.S3
$LifecycleRule = [Amazon.S3.Model.LifecycleRule]::new()
```

For more information, see the version 4 announcement for the AWS Tools for PowerShell.

(back to top (p. 67))
Supported browsers for AWS CloudShell

The following table lists the supported browsers for AWS CloudShell.

**Web browser support**

<table>
<thead>
<tr>
<th>Browser</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Chrome</td>
<td>Latest three major versions</td>
</tr>
<tr>
<td>Mozilla Firefox</td>
<td>Latest three major versions</td>
</tr>
<tr>
<td>Microsoft Edge</td>
<td>Latest three major versions</td>
</tr>
<tr>
<td>Apple Safari for macOS</td>
<td>Latest two major versions</td>
</tr>
</tbody>
</table>
Supported AWS Regions for AWS CloudShell

This section covers the list of supported AWS Regions and Opt-in Regions for AWS CloudShell. For a list of AWS service endpoints and quotas for CloudShell, see the AWS CloudShell page in the Amazon Web Services General Reference.

The following are the supported AWS Regions for AWS CloudShell:

- US East (Ohio)
- US East (N. Virginia)
- US West (N. California)
- US West (Oregon)
- Asia Pacific (Mumbai)
- Asia Pacific (Osaka)
- Asia Pacific (Seoul)
- Asia Pacific (Sydney)
- Asia Pacific (Singapore)
- Asia Pacific (Tokyo)
- Canada (Central)
- Europe (Frankfurt)
- Europe (Ireland)
- Europe (London)
- Europe (Paris)
- Europe (Stockholm)
- South America (São Paulo)

GovCloud Regions

The following are the supported GovCloud Regions for CloudShell:

- AWS GovCloud (US-East)
- AWS GovCloud (US-West)

Opt-in Regions

Opt-in Regions aren't enabled by default. You must manually enable these Regions to use them. For more information, see Managing AWS Regions. The following are the supported Opt-in Regions for CloudShell:

- Africa (Cape Town)
- Asia Pacific (Hong Kong)
- Asia Pacific (Jakarta)
• Europe (Milan)
• Middle East (Bahrain)
• Middle East (UAE)
Service quotas and restrictions for AWS CloudShell

This page describes the Service quotas and restrictions that apply to the following areas:

- Persistent storage (p. 74)
- Monthly usage (p. 74)
- Command size (p. 75)
- Concurrent shells (p. 75)
- Shell sessions (p. 75)
- Network access and data transfer (p. 75)
- System files and page reloads (p. 75)

Persistent storage

With AWS CloudShell, you have persistent storage of 1 GB for each AWS Region at no cost. Persistent storage is located in your home directory ($HOME) and is private to you. Unlike ephemeral environment resources that are recycled after each shell session ends, data in your home directory persists between sessions.

If you stop using AWS CloudShell in an AWS Region, data is retained in the persistent storage of that Region for 120 days after the end of your last session. After 120 days, unless you take action, your data is automatically deleted from the persistent storage of that Region. You can prevent removal by launching AWS CloudShell again in that AWS Region. For more information, see Step 2: Select a Region, launch AWS CloudShell, and choose a shell (p. 14).

Note

Usage scenario
Márcia has used AWS CloudShell to store files in her home directories in two AWS Regions: US East (N. Virginia) and Europe (Ireland). She then started using AWS CloudShell exclusively in Europe (Ireland) and stopped launching shell sessions in US East (N. Virginia).
Before the deadline for deleting data in US East (N. Virginia), Márcia decides to prevent her home directory from being recycled by launching AWS CloudShell and selecting the US East (N. Virginia) Region again. Because she has continually used Europe (Ireland) for shell sessions, her persistent storage in that Region isn't affected.

Monthly usage

There are monthly usage quotas for AWS CloudShell for each AWS Region in your AWS account. If you attempt to access AWS CloudShell after you reached the monthly quota for that Region, a message displays to explain why the shell environment can't be started.

Note
If you need to increase your monthly usage quotas, contact AWS Support.
Command size

The command size cannot exceed 65412 characters.

**Note**
If you intend to execute the command that exceeds 65412 characters, then create a script with the language of your choice, and then execute it from the command line interface. For more information about the range of pre-installed software that can be accessed from the command line interface, see Pre-installed software (p. 56).
To see as an example of how to create a script, and then execute it from the command line interface, see Tutorial: Getting started with AWS CloudShell (p. 11).

Concurrent shells

- **Concurrent shells**: You can run up to 10 shells at the same time in each AWS Region for your account.

Shell sessions

- **Inactive sessions**: AWS CloudShell is an interactive shell environment—if you don't interact with it using your keyboard or pointer for 20–30 minutes, your shell session ends. Running processes don't count as interactions.
- **Long-running sessions**: A shell session that runs continuously for approximately 12 hours automatically end even if the user is regularly interacting with it during that period.

Network access and data transfer

The following restrictions apply to both the inbound and outbound traffic of your AWS CloudShell environment:

- **Outbound**: You can access the public internet.
- **Inbound**: You can't access inbound ports. No public IP address is available.

**Warning**
With access to the public internet, there's a risk that certain users might export data from the AWS CloudShell environment. We recommend that IAM administrators manage the allow list of trusted AWS CloudShell users through IAM tools. For information about how specific users can be explicitly denied access, see Managing allowable actions in AWS CloudShell using custom policies (p. 44).

*Data transfer*: Uploading and downloading files to and from AWS CloudShell might be slow for large files. Alternatively, you can transfer files to your environment from an Amazon S3 bucket using the command line interface of the shell.

Restrictions on system files and page reloads

- **System files**: If you incorrectly modify files that are required by the compute environment, you might experience problems when accessing or using the AWS CloudShell environment. If this occurs, you might need to deleting your home directory (p. 19) to regain access.
• **Reloading pages**: To reload the AWS CloudShell interface, use the refresh button in your browser instead of the default shortcut key sequence for your operating system.
## Document history for the AWS CloudShell User Guide

### Recent updates

The following table describes important changes to the *AWS CloudShell User Guide*.

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
</table>
| **New AWS Regions for AWS CloudShell (p. 77)** | AWS CloudShell is now generally available in the following AWS Regions:  
- US West (N. California)  
- Africa (Cape Town)  
- Asia Pacific (Hong Kong)  
- Asia Pacific (Osaka)  
- Asia Pacific (Seoul)  
- Asia Pacific (Jakarta)  
- Asia Pacific (Singapore)  
- Europe (Paris)  
- Europe (Stockholm)  
- Europe (Milan)  
- Middle East (Bahrain)  
- Middle East (UAE) | June 16, 2023 |
| **Launch AWS CloudShell on the Console Toolbar (p. 77)** | Launch CloudShell on the Console Toolbar, on the lower left of the console by choosing CloudShell. | March 28, 2023 |
| **New AWS Regions for AWS CloudShell (p. 77)** | AWS CloudShell is now available in the following AWS Regions:  
- Canada (Central)  
- Europe (London)  
- South America (São Paulo) | October 6, 2022 |
<p>| <strong>AWS CloudShell supported in US AWS GovCloud (p. 77)</strong> | AWS CloudShell is now supported in the AWS GovCloud (US) Region. | June 29, 2022 |
| <strong>Security FAQs (p. 53)</strong> | Additional FAQs focused on security issues. | April 14, 2022 |
| <strong>Web Sockets (p. 55)</strong> | Added section to network requirements that explains CloudShell's use of the WebSocket protocol. | March 21, 2022 |</p>
<table>
<thead>
<tr>
<th>Topic</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Troubleshooting arrow keys in PowerShell (p. 68)</strong></td>
<td>Follow the steps to fix arrow keys that incorrectly output letters when pressed.</td>
<td>February 7, 2022</td>
</tr>
<tr>
<td><strong>Tab key autocomplete (p. 77)</strong></td>
<td>New documentation that explains how to use bash-completion, which allows the autocompletion of partially typed commands or arguments by pressing the Tab key.</td>
<td>September 24, 2021</td>
</tr>
<tr>
<td><strong>Specifying AWS Regions (p. 32)</strong></td>
<td>Documentation on specifying default AWS Region for AWS CLI commands.</td>
<td>May 11, 2021</td>
</tr>
<tr>
<td><strong>Formatting in PDF and Kindle versions (p. 77)</strong></td>
<td>Fixed image sizes and text in table cells.</td>
<td>March 10, 2021</td>
</tr>
</tbody>
</table>
| **General availability (GA) release of AWS CloudShell in selected AWS Regions (p. 77)** | AWS CloudShell is now generally available in the following AWS Regions:  
  - US East (Ohio)  
  - US East (N. Virginia)  
  - US West (Oregon)  
  - Asia Pacific (Tokyo)  
  - Europe (Ireland)  
  - Asia Pacific (Mumbai)  
  - Asia Pacific (Sydney)  
  - Europe (Frankfurt) | December 15, 2020 |