User Guide

AWS CloudShell

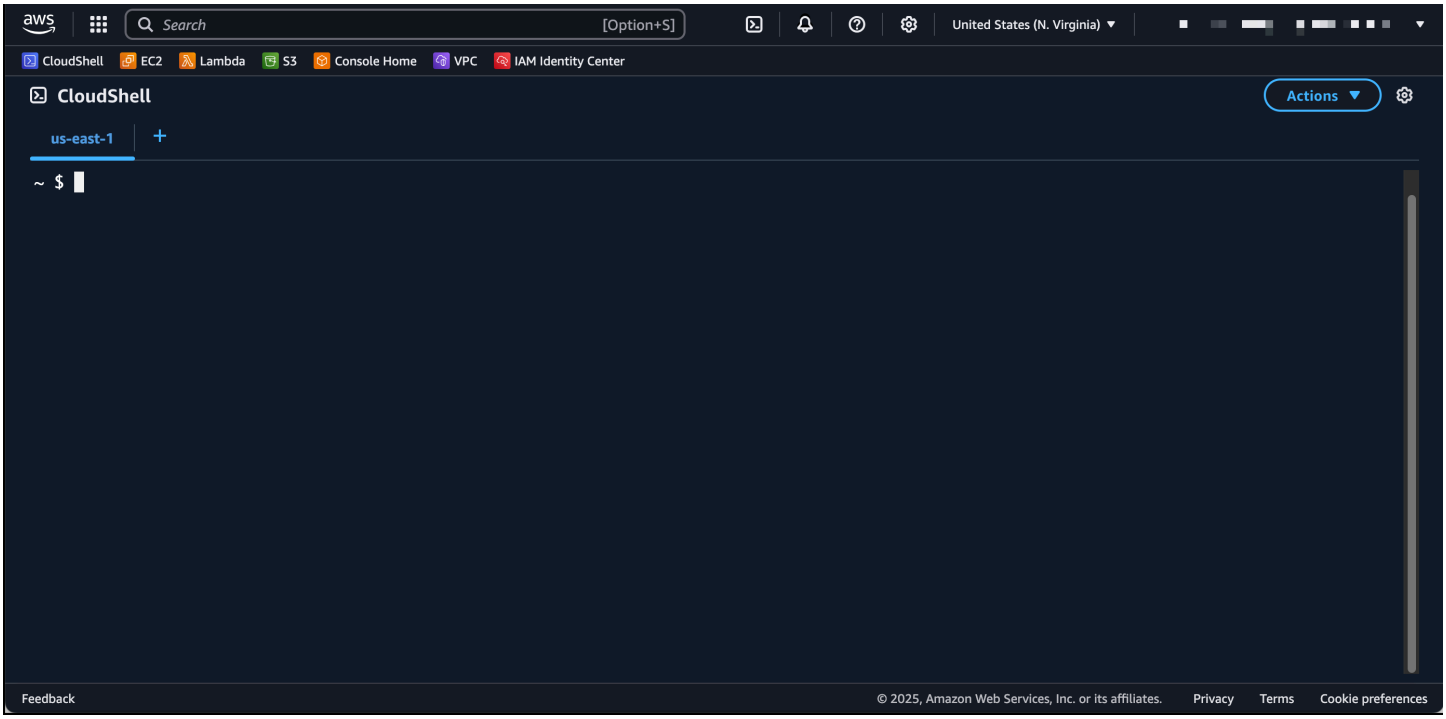
AWS CloudShell: User Guide

Table of Contents

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 Getting started with AWS CloudShell

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 that's based on Amazon Linux 2023 is created. Within this environment, you can access an extensive range of pre-installed development tools, options for uploading and downloading files, and file storage that persists between sessions. You can use CloudShell in the most recent versions of Google Chrome, Mozilla Firefox, Microsoft Edge, and Apple Safari browsers.

(Try it now: Getting started with AWS CloudShell)

Features of AWS CloudShell

AWS CloudShell provides the following 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 Manage AWS services from CLI in CloudShell.

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.

For more information, see AWS CloudShell compute environment: specifications and software.

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.

> **Note**
>
> CloudShell VPC environments do not have persistent storage. The $HOME directory is deleted when your VPC environment times out (after 20-30 minutes of inactivity), or when you delete or restart your environment.

CloudShell VPC environments

AWS CloudShell virtual private cloud (VPC) enables you to create a CloudShell environment in your VPC. For each VPC environment, you can assign a VPC, add a subnet, and associate one or more security groups. AWS CloudShell inherits the network configuration of the VPC and enables you to use AWS CloudShell securely within the same subnet as other resources in the VPC.

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.

Shell session management

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

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.

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.

You can also extend your shell environment by installing your own software and modifying your shell with scripts.

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.

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.

Key AWS CloudShell topics

- Getting started with AWS CloudShell
- AWS CloudShell Concepts
- Manage AWS services from CLI in CloudShell
- Customizing your AWS CloudShell experience
- AWS CloudShell compute environment: specifications and software

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.

```
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

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.

- 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**.

  Main sign-in page

  - Open https://aws.amazon.com/console/.

- 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**.
- 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**.

> **Note**
>
> You can also sign in as a root user. This identity has complete access to all AWS services and resources in the account. We strongly recommend that you don't use the root user for everyday tasks, even administrative ones. Instead, adhere to the best practice of using the root user only to create your first IAM user.

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

In this step, you launch 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 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.

Example

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.
      - You can adjust the height of your CloudShell session by dragging =.
      - You can switch your CloudShell session to a full screen by clicking **Open in new browser tab**.

   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.

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 in the AWS CloudShell compute environment section.

Step 3: Download a file from AWS CloudShell

> **Note**
>
> This option is not available for VPC environments.

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.

> **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 `mydownload-file`, 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.

Step 4: Upload a file to AWS CloudShell

> **Note**
>
> This option is not available for VPC environments.

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.

   ```
   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.

Step 6: Create a home directory backup

This step describes how to create a home directory backup.

1. **Create a backup file**

   Create a temporary folder outside the home directory.

   ```
   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:

      ```
      tar \
          --create \
          --gzip \
          --verbose \
          --file=${HOME_BACKUP_DIR}/home.tar.gz \
          [--exclude ${HOME}/.cache] \ // Optional
          ${HOME}/
      echo "Home directory backed up to this file: ${HOME_BACKUP_DIR}/home.tar.gz"
      ```

   b. **Create a backup file using zip**

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

      ```
      zip \
          --recurse-paths \
          ${HOME_BACKUP_DIR}/home.zip \
          ${HOME} \
          [--exclude ${HOME}/.cache/\*] // Optional
      ```

```
echo "Home directory backed up to this file: ${HOME_BACKUP_DIR}/home.zip"
```

2. **Transfer the backup file outside CloudShell**

   You can use one of the following options to transfer the backup file outside CloudShell:

   a. **Download the backup file on your local machine**

      You can download the file created in the previous step. For more information about how to download a file from CloudShell, see Download a file from AWS CloudShell.

      In the download file dialogue box, enter the path for the file to be downloaded (for example, `/tmp/tmp.iA99tD9L98/home.tar.gz`).

   b. **Transfer the backup file to S3**

      To generate a bucket, enter the following command:

      ```
      aws s3 mb s3://${BUCKET_NAME}
      ```

      Use AWS CLI to copy the file to the S3 bucket:

      ```
      aws s3 cp ${HOME_BACKUP_DIR}/home.tar.gz s3://${BUCKET_NAME}
      ```

      > **Note**
      >
      > Data transfer charges might apply.

3. **Backup directly to an S3 bucket**

   To backup directly to an S3 bucket, enter the following command:

   ```
   aws s3 cp \
       ${HOME}/ \
       s3://${BUCKET_NAME} \
       --recursive \
       [--exclude .cache/\*] // Optional
   ```

Step 7: Restart a shell session

This step describes how to 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.

1. To restart a shell session, choose **Actions**, **Restart**.

   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

This step describes how to delete a shell session.

> **Note**
>
> This option is not available for VPC environments. When you restart a VPC environment, its home directory is deleted.

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 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.

1. To delete a shell session, choose **Actions**, **Delete**.

   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**.

   AWS CloudShell stops all active sessions in the current AWS Region. You can create a new environment or setup a CloudShell VPC environment.

3. To create a new environment, choose **Open a tab**.
4. To create a CloudShell VPC environment, choose **Create a VPC environment**.

Manually exiting shell sessions

With the command line, you can leave a shell session and log out using the `exit` command. You can then press any key to reconnect and continue to use AWS CloudShell.

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.

   ```
   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 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.

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**
>
> 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:

   ```
   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.
   >
   > ```
   > $ 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.

```
{
    "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.

```
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:

```
{"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

- Manage AWS services from CLI in CloudShell
- Copying multiple files between your local machine and CloudShell
- AWS CloudShell Concepts
- Customizing your AWS CloudShell experience

AWS CloudShell tutorials

The following tutorials show you how to experiment, and test different functionalities and integrations when using AWS CloudShell.

Tutorial overview	Learn more
Copying multiple files	the section called “Tutorial: Copying multiple files”
Creating presigned URLs	???
Building a Docker container inside AWS CloudShell and pushing to Amazon ECR	???
Deploying a Lambda function using the AWS CDK	???

Copying multiple files between your local machine and CloudShell

This tutorial shows how to copy multiple files between your local machine and CloudShell.

Using the AWS 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

This step describes how to upload and download 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`
- `s3:ListBucket`

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

This step describes how to upload multiple files 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:

   ```
   {
       "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/.

     (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? 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.

- 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:

```
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, download messages are displayed for every file downloaded from the bucket to the directory.

> **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

This step describes how to download multiple files 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

This step describes how to upload and download 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.

Upload multiple files to AWS CloudShell using zipped folders

This step describes how to upload multiple files 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

This step describes how to download multiple files 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.

Creating a presigned URL for Amazon S3 objects using 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 in the *Amazon Simple Storage Service User Guide*.

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

This step describes how to 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": "AROAXXOZUUOTTWDCVIDZ2: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 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 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

This step describes how to 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 = "?Action=login"
  request_parameters += "&Issuer=Example.org"
  request_parameters += "&Destination=" + urllib.parse.quote("https://us-
west-2.console.aws.amazon.com/cloudshell")
```

```
  request_parameters += "&SigninToken=" + signin_token["SigninToken"]
  request_url = "https://signin.aws.amazon.com/federation" + request_parameters

  # Send final URL to stdout
  print (request_url)

if __name__ == "__main__":
  main()
```

2. Save the code in a file called `share.py`.
3. Run the following from the command line to retrieve the Amazon Resource Name (ARN) of the IAM role from CloudFormation. Then, use it in the Python script to obtain temporary security credentials.

```
ROLE_ARN=$(aws cloudformation describe-stacks --stack-name CollaboratorRole --query
 "Stacks[*].Outputs[?OutputKey=='CollaboratorRoleArn'].OutputValue" --output text)
 python3 ./share.py
```

The script returns a URL that a collaborator can click to take them to AWS CloudShell in AWS Management Console. The collaborator has full control over the `myfolder/` folder in the Amazon S3 bucket for the next 3,600 seconds (1 hour). The credentials expire after an hour. After this time, the collaborator can no longer access the bucket.

Building a Docker container inside CloudShell and pushing it to an Amazon ECR repository

This tutorial shows you how to define and build a Docker container in AWS CloudShell and push it to an Amazon ECR repository.

Prerequisites

- You must have the necessary permissions to create and push to an Amazon ECR repository. For more information about repositories with Amazon ECR, see Amazon ECR private repositories in the *Amazon ECR User Guide*. For more information about the permissions required for pushing images with Amazon ECR, see Required IAM permissions for pushing an image in the *Amazon ECR User Guide*.

Tutorial procedure

The following tutorial outlines how to use the CloudShell interface to build a Docker container and push it to an Amazon ECR repository.

1. Create a new folder in your home directory.

```
mkdir ~/docker-cli-tutorial
```

2. Navigate to the folder you created.

```
cd ~/docker-cli-tutorial
```

3. Create an empty Dockerfile.

```
touch Dockerfile
```

4. Using a text editor, for example `nano Dockerfile`, open the file and paste the following content into it.

```
# Dockerfile

# Base this container on the latest Amazon Linux version
FROM public.ecr.aws/amazonlinux/amazonlinux:latest

# Install the cowsay binary
RUN dnf install --assumeyes cowsay

# Default entrypoint binary
ENTRYPOINT [ "cowsay" ]

# Default argument for the cowsay entrypoint
CMD [ "Hello, World!" ]
```

5. The Dockerfile is now ready to be built. Build the container by running `docker build`. Tag the container with an easy-to-type name for use in future commands.

```
docker build --tag test-container .
```

Make sure to include the trailing period (`.`).
Image of the docker build command run inside AWS CloudShell.

6. You can now test the container to check that it is running correctly in AWS CloudShell.

```
docker container run test-container
```

Image of the docker container run command inside AWS CloudShell

7. Now that you have a functioning Docker container, you need to push it to an Amazon ECR repository. If you have an existing Amazon ECR repository, you can skip this step.

Run the following command to create an Amazon ECR repository for this tutorial.

```
ECR_REPO_NAME=docker-tutorial-repo
aws ecr create-repository --repository-name ${ECR_REPO_NAME}
```

Image of the command used to create an Amazon ECR repository inside AWS CloudShell

8. After you create the Amazon ECR repository, you can push the Docker container to it.

Run the following command to get the Amazon ECR sign-in credentials for Docker.

```
AWS_ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
ECR_URL=${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com
aws ecr get-login-password | docker login --username AWS --password-stdin
 ${ECR_URL}
```

Image of the command used to get the Amazon ECR sign-in credentials for Docker.

> **Note**
>
> If the **AWS_REGION** environment variable is not set in your CloudShell or you want to interact with resources in other AWS Regions, run the following command:
>
> ```
> AWS_REGION=<your-desired-region>
> ```

9. Tag the image with the target Amazon ECR repository and then push it to that repository.

```
docker tag test-container ${ECR_URL}/${ECR_REPO_NAME}
docker push ${ECR_URL}/${ECR_REPO_NAME}
```

Image of the command used to tag the image with the target Amazon ECR repository.

If you encounter errors or run into issues when trying to complete this tutorial, see the Troubleshooting section of this guide for help.

Clean up

You have now successfully deployed your Docker container to your Amazon ECR repository. To remove the files you created in this tutorial from your AWS CloudShell environment, run the following command.

- ```
  cd ~
  rm -rf ~/docker-cli-tutorial
  ```
- Delete the Amazon ECR repository.

  ```
  aws ecr delete-repository --force --repository-name ${ECR_REPO_NAME}
  ```

Deploying a Lambda function using the AWS CDK in CloudShell

This tutorial shows you how to deploy a Lambda function to your account using the AWS Cloud Development Kit (AWS CDK) in CloudShell.

Prerequisites

- Bootstrap your account for use with the AWS CDK. For information about bootstrapping with AWS CDK, see Bootstrapping in the *AWS CDK v2 Developer Guide*. If you haven't bootstrapped the account, you can run `cdk bootstrap` in CloudShell.

- Make sure you have the appropriate permissions to deploy resources to your account. Administrator permissions are recommended.

Tutorial procedure

The following tutorial outlines how to deploy a Docker container-based Lambda function using the AWS CDK in CloudShell.

1. Create a new folder in your home directory.
  ```

```
mkdir ~/docker-cdk-tutorial
```

2. Navigate to the folder you created.

```
cd ~/docker-cdk-tutorial
```

3. Install the AWS CDK dependencies locally.

```
npm install aws-cdk aws-cdk-lib
```

Image of the command used to install the AWS CDK dependencies.

4. Create a skeleton AWS CDK project in the folder that you created.

```
touch cdk.json
mkdir lib
touch lib/docker-tutorial.js lib/Dockerfile lib/hello.js
```

5. Using a text editor, for example `nano cdk.json`, open the file and paste the following content into it.

```
{
  "app": "node lib/docker-tutorial.js"
}
```

6. Open the `lib/docker-tutorial.js` file and paste the following content into it.

```
// this file defines the CDK constructs we want to deploy

const { App, Stack } = require('aws-cdk-lib');
const { DockerImageFunction, DockerImageCode } = require('aws-cdk-lib/aws-lambda');
const path = require('path');

// create an application
const app = new App();

// define stack
class DockerTutorialStack extends Stack {
  constructor(scope, id, props) {
    super(scope, id, props);

    // define lambda that uses a Docker container
```

```
    const dockerfileDir = path.join(__dirname);
    new DockerImageFunction(this, 'DockerTutorialFunction', {
      code: DockerImageCode.fromImageAsset(dockerfileDir),
      functionName: 'DockerTutorialFunction',
    });
  }
}

// instantiate stack
new DockerTutorialStack(app, 'DockerTutorialStack');
```

7. Open the `lib/Dockerfile` and paste the following content into it.

```
# Use a NodeJS 20.x runtime
FROM public.ecr.aws/lambda/nodejs:20

# Copy the function code to the LAMBDA_TASK_ROOT directory
# This environment variable is provided by the lambda base image
COPY hello.js ${LAMBDA_TASK_ROOT}

# Set the CMD to the function handler
CMD [ "hello.handler" ]
```

8. Open the `lib/hello.js` file and paste the following content into it.

```
// define the handler
exports.handler = async (event) => {
  // simply return a friendly success response
  const response = {
    statusCode: 200,
    body: JSON.stringify('Hello, World!'),
  };
  return response;
};
```

9. Use the AWS CDK CLI to synthesize the project and deploy the resources. You must bootstrap your account.

```
npx cdk synth
npx cdk deploy --require-approval never
```

Image of the command to use the AWS CDK CLI to synthesize the project and deploy the resources.

10. Invoke the Lambda function to confirm and verify it.

```
aws lambda invoke --function-name DockerTutorialFunction out.json
jq . out.json
```

Image of the command used to invoke the Lambda function.

You have now successfully deployed a Docker container-based Lambda function using the AWS CDK. For more information on AWS CDK, see the AWS CDKv2 Developer Guide. If you encounter errors or run into issues when trying to complete this tutorial, see the Troubleshooting section of this guide for help.

Clean up

You have now successfully deployed a Docker container-based Lambda function using the AWS CDK. Inside the AWS CDK project, run the following command to delete the associated resources. You will be prompted to confirm the deletion.

-
```
npx cdk destroy DockerTutorialStack
```

- To remove the files and resources you created in this tutorial from your AWS CloudShell environment, run the following command.

```
cd ~
rm -rf ~/docker-cli-tutorial
```

AWS CloudShell Concepts

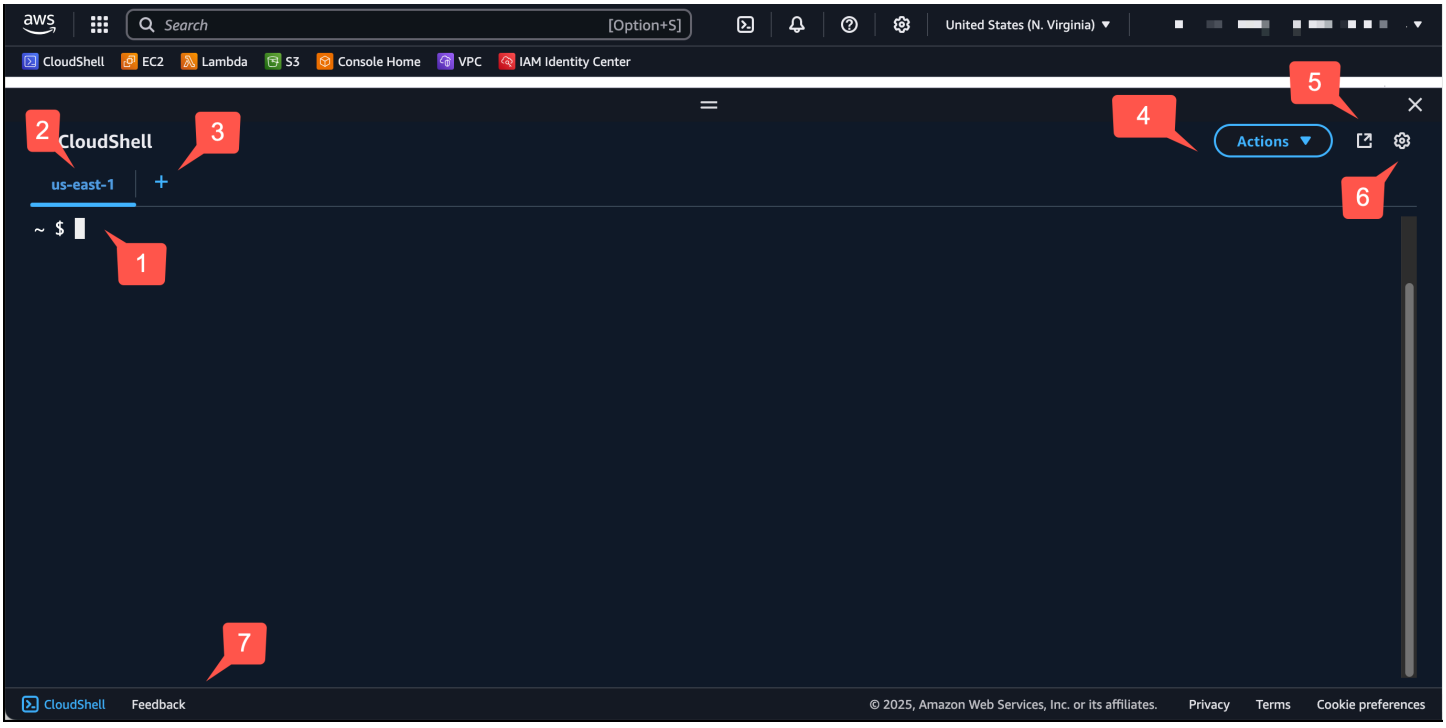
This section describes how to interact with AWS CloudShell and perform specific actions with supported applications.

Topics

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. 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 **+** icon is a dropdown menu that includes options to create, restart, and delete environments.
4. The **Actions** menu, which provides options for changing the screen layout, downloading and uploading files, restarting your AWS CloudShell, and deleting your AWS CloudShell home directory.

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

5. The **Open in new browser tab**, which provides the option to access your CloudShell session in a full screen.
6. The **Preferences** option, which you can use to customize your shell experience.
7. 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:
       - From the console, launch CloudShell, and choose **Feedback**. Add your comments, and then choose **Submit**.
       - Choose **CloudShell** on the Console Toolbar, on the lower left of the console, and then choose **Open in new browser tab** icon, **Feedback**. Add your comments, and then choose **Submit**.

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

   - Learn about our privacy policy and terms of use, and customize cookie preferences.

Working in AWS Regions

The current AWS Region that you're running in is displayed as a tab.

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.
- Persistent storage is not available for AWS CloudShell VPC environments.

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

```
$ 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

```
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.

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.

Access CloudShell in the Console Mobile Application

You can access CloudShell in the AWS Console Mobile Application from the home screen. From the home screen, you can view information about CloudShell and other AWS services. For more information, see Getting started with the AWS Console Mobile Application. To launch CloudShell in the AWS Console Mobile Application, choose one of the following options:

- Select the **CloudShell** icon at the bottom of the navigation bar.
- Select the **CloudShell** on the Services menu.

You can exit CloudShell at any time by choosing **X**.

For more information about accessing CloudShell in Console Mobile Application, see Access AWS CloudShell.

Note

Currently, you cannot create or launch VPC environments in the AWS Console Mobile Application.

Working with Docker

AWS CloudShell fully supports Docker without installation or configuration. You can define, build and run Docker containers inside AWS CloudShell. You can deploy Docker-based resources, such as Lambda functions based on Docker containers, via the AWS CDK Toolkit as well as build Docker containers and push them to Amazon ECR repositories via the Docker CLI. For detailed steps on how to run both of these deployments, see the following tutorials:

- Tutorial: Deploying a Lambda function using the AWS CDK

- Tutorial: Building a Docker container inside AWS CloudShell and pushing it to an Amazon ECR repository

There are certain restrictions and limitations with using Docker with AWS CloudShell:

- Docker has limited space in an environment. If you have large individual images, or too many pre-existing Docker images, it can cause issues that might prevent you from pulling, building, or running additional images. For more information on Docker, see the Docker Documentation guide.
- Docker is available in all AWS Regions, except the AWS GovCloud (US) Regions. For a list of Regions in which Docker is available, see Supported AWS Regions for AWS CloudShell.
- If you encounter issues when using Docker with AWS CloudShell, see the Troubleshooting section of this guide for information on how to potentially resolve these issues.

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”.
- **Theme** – Choose between **Light** and **Dark** interface themes. For more information about changing the interface theme, see the section called “Changing the interface theme”.

Manage AWS services from CLI in 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
- AWS Elastic Beanstalk CLI
- Amazon ECS CLI
- AWS SAM

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.)

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
- Amazon EC2
- Amazon Glacier

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 \
    --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*.

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:

```
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*.

Amazon Glacier

Amazon Glacier and Amazon 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:

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

For more information, see Using Amazon 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 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 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 python3.9 --dependency-manager pip --app-template hello-world --name
 sam-app
```

For more information, see the AWS SAM CLI command reference in the *AWS Serverless Application Model Developer Guide*.

Using Kiro CLI in CloudShell

The Kiro CLI is a command-line interface that allows you to interact with Kiro. For more information, see Core Features of Kiro CLI in the *Kiro User Guide*.

Kiro CLI in CloudShell allows you to interact in natural language conversations, ask questions, and receive responses from Kiro all from your terminal. You can get the related shell command that reduces the need to search for, remember syntax, and receive command suggestions as you type in the terminal.

If you don't see Kiro CLI features in CloudShell, contact your administrator to provide you IAM permissions. For more information, see Identity-based policy examples for Kiro Developer in the *Kiro User Guide*.

This chapter explains how you can use Kiro CLI features in CloudShell.

Some Kiro CLI features require authentication. For more information, see Authentication in the *Kiro User Guide*.

Using Kiro chat command in CloudShell

The `kiro-cli` command allows you to ask questions and receive responses from Kiro all from your terminal. To initiate a conversation with Kiro, run `kiro-cli` command in the CloudShell terminal. For more information, see Chatting with Kiro in the CLI in the *Kiro User Guide*.

Using Kiro translate command in CloudShell

The `kiro-cli translate` command allows you to write natural language instruction. To translate with Kiro, run `kiro-cli translate` command in the CloudShell terminal. For more information, see Translating from natural language to bash in the *Kiro User Guide*.

CLI command completion in CloudShell

CLI completion in CloudShell provides suggestions for commands and options as you type in the terminal. For more information, see Generating command line completion in *Kiro User Guide*.

Using Kiro inline suggestions in CloudShell

The Kiro inline suggestions in CloudShell provide you with command suggestions as you type in the terminal. For more information, see Kiro inline on the command line in the *Kiro User Guide*.

To use Kiro inline suggestions in CloudShell

1. From the AWS Management Console, Choose **CloudShell**.
2. On the CloudShell terminal, switch to Z shell, and start typing. To switch to Z shell, type `zsh` in the terminal, and then press **Enter**.

   > **Note**
   >
   > Currently, Kiro inline is only supported in Z shell.

   When you start typing your command, Kiro will make suggestions based on your current input and previous commands. Inline suggestions are automatically enabled.

To disable the inline suggestions, run the following command:

```
kiro-cli inline disable
```

To enable the inline suggestions, run the following command:

```
kiro-cli inline enable
```

Identity-based policy for Kiro CLI in CloudShell

To use Kiro CLI in CloudShell, make sure you have the required IAM permissions. For more information, see Identity-based policy examples for Kiro Developer in the *Kiro User Guide*.

Running a command in CloudShell from AWS Service consoles

You can run a command in the CloudShell terminal through Amazon ElastiCache and Amazon DocumentDB (with MongoDB compatibility) consoles in the AWS Management Console.

To run a command in CloudShell from other AWS Service consoles, the IAM policy assigned to your role must include `cloudshell:approveCommand` permissions.

CloudShell opens on the **Console Toolbar** and **Run command** pop-up appears in CloudShell. On the **Run command** pop-up, the command appears in the command box.

To run a command in the CloudShell terminal, choose one of the following steps:

1. Enter a name in the **New environment name** box if you have not created a VPC environment in the CloudShell.

   You can view the VPC environment details that is based on the VPC details of your resource.

   a. Choose **Create and run**.

      This step will create a new CloudShell VPC environment and run the command in the CloudShell terminal.

2. You can view the CloudShell environment name if you have already created a CloudShell VPC environment.

   > **Note**
   >
   > If you already have a CloudShell VPC environment, you can't create a new VPC environment.

   a. Choose **Run**.

      This step will run the command in the CloudShell terminal in the selected CloudShell VPC environment.

> **Note**
>
> If you don't have permission to view the created VPC environments, contact your administrator to add the `cloudshell:describeEnvironments` permission. For more information, see Managing AWS CloudShell access and usage with IAM policies.

You can continue to run commands in the CloudShell terminal.

Customizing your AWS CloudShell experience

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

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

You can also extend your shell environment by installing your own software and modifying your shell with scripts.

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

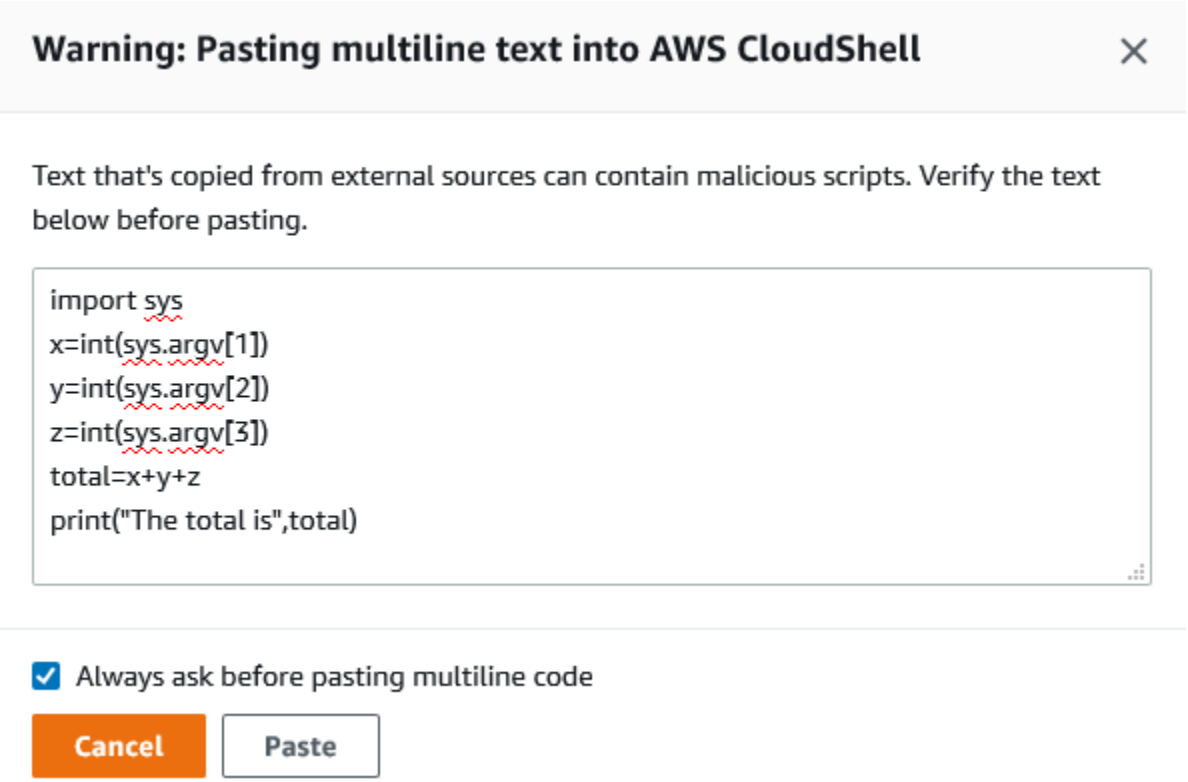
Toggle between light and dark theme for the command line interface.

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**.



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.

Using Amazon Q CLI

You can enable or disable Amazon Q CLI by choosing **Preferences**, **Enable Amazon Q CLI** and **Disable Amazon Q CLI**. For more information, see Enable/disable Amazon Q CLI.

Using AWS CloudShell in Amazon VPC

AWS CloudShell virtual private cloud (VPC) enables you to create a CloudShell environment in your VPC. For each VPC environment, you can assign a VPC, add a subnet, and associate up to five security groups. AWS CloudShell inherits the network configuration of the VPC and enables you to use AWS CloudShell securely within the same subnet as other resources in the VPC and connect to them.

With Amazon VPC, you can launch AWS resources in a logically isolated virtual network that you've defined. This virtual network closely resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS. For more information about VPC, see Amazon Virtual Private Cloud.

Operating constraints

AWS CloudShell VPC environments have the following constraints:

- You can create a maximum of two VPC environments per IAM principal.
- You can assign a maximum of five security groups for a VPC environment.
- You cannot use the CloudShell upload and download options in the Actions menu for VPC environments.

> **Note**
>
> It is possible to upload or download files from VPC environments that have access to the internet ingress/egress through other CLI tools.

- VPC environments do not support persistent storage. Storage is ephemeral. Data and home directory are deleted when an active environment session ends.
- Your AWS CloudShell environment can only connect to the internet if it is in a private VPC subnet.

> **Note**
>
> Public IP addresses are not allocated to CloudShell VPC environments by default. VPC environments created in public subnets with routing tables configured to route all traffic to Internet Gateway will not have access to public internet, but private subnets

configured with Network Address Translation (NAT) have access to public internet. VPC environments created in such private subnets will have access to public internet.

- To provide a managed CloudShell environment for your account, AWS might provision network access to the following services for the underlying compute host:
  - Amazon S3
  - VPC endpoints
    - com.amazonaws.<region>.ssmmessages
    - com.amazonaws.<region>.logs
    - com.amazonaws.<region>.kms
    - com.amazonaws.<region>.execute-api
    - com.amazonaws.<region>.ecs-telemetry
    - com.amazonaws.<region>.ecs-agent
    - com.amazonaws.<region>.ecs
    - com.amazonaws.<region>.ecr.dkr
    - com.amazonaws.<region>.ecr.api
    - com.amazonaws.<region>.codecatalyst.packages
    - com.amazonaws.<region>.codecatalyst.git
    - aws.api.global.codecatalyst

  You cannot restrict access to these endpoints by modifying your VPC configuration.

  CloudShell VPC is available in all AWS Regions and GovCloud Regions. For a list of Regions in which CloudShell VPC is available, see Supported AWS Regions for AWS CloudShell.

Creating a CloudShell VPC environment

This topic walks you through the steps to create a VPC environment in CloudShell.

Prerequisites

Your administrator must provide the necessary IAM permissions for you to be able to create VPC environments. For more information about enabling permissions to create CloudShell VPC environments, see the section called “Required IAM permissions for creating and using CloudShell VPC environments”.

To create a CloudShell VPC environment

1. On the CloudShell console page, choose the **+** icon and then choose **Create VPC environment** from the dropdown menu.
2. On the **Create a VPC environment** page, enter a name for your VPC environment in the **Name** box.
3. From the **Virtual private cloud (VPC)** dropdown list, choose a VPC.
4. From the **Subnet** dropdown list, choose a subnet.
5. From the **Security** group dropdown list, choose one or more security groups that you want to assign to your VPC environment.

   > **Note**
   >
   > You can choose a maximum of five security groups.

6. Choose **Create** to create your VPC environment.
7. (Optional) Choose **Actions**, and then choose **View details** to review the details of the newly created VPC environment. The IP address of your VPC environment is displayed in the command line prompt.

For information about using VPC environments, see *Getting started*.

Required IAM permissions for creating and using CloudShell VPC environments

To create and use CloudShell VPC environments, the IAM administrator must enable access to VPC specific Amazon EC2 permissions. This section lists the Amazon EC2 permissions needed to create and use VPC environments.

To create VPC environments, the IAM policy assigned to your role must include the following Amazon EC2 permissions:

- `ec2:DescribeVpcs`
- `ec2:DescribeSubnets`
- `ec2:DescribeSecurityGroups`
- `ec2:DescribeDhcpOptions`

- `ec2:DescribeNetworkInterfaces`

- `ec2:CreateTags`
- `ec2:CreateNetworkInterface`
- `ec2:CreateNetworkInterfacePermission`

We recommend to include:

- **ec2:DeleteNetworkInterface**

> **Note**
>
> This permission is not mandatory, but this is required for CloudShell to clean up the ENI resource (ENIs created for CloudShell VPC environments are tagged with **ManagedByCloudShell** key) created by it. If this permission not in enabled, you must manually clean up the ENI resource after every CloudShell VPC environment use.

IAM policy granting full CloudShell access including access to VPC

The following example displays how to enable full permissions, including access to VPC, to CloudShell:

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Sid": "AllowCloudShellOperations",
   "Effect": "Allow",
   "Action": [
    "cloudshell:*"
   ],
   "Resource": "*"
  },
  {
   "Sid": "AllowDescribeVPC",
```

```
    "Effect": "Allow",
    "Action": [
     "ec2:DescribeSubnets",
     "ec2:DescribeSecurityGroups",
     "ec2:DescribeVpcs"
    ],
    "Resource": "*"
   },
   {
    "Sid": "AllowInspectVPCConfigurationViaCloudShell",
    "Effect": "Allow",
    "Action": [
     "ec2:DescribeDhcpOptions",
     "ec2:DescribeNetworkInterfaces"
    ],
    "Resource": "*",
    "Condition": {
     "ForAnyValue:StringEquals": {
      "aws:CalledVia": "cloudshell.amazonaws.com"
     }
    }
   },
   {
    "Sid": "AllowCreateTagWithCloudShellKeyViaCloudShell",
    "Effect": "Allow",
    "Action": [
     "ec2:CreateTags"
    ],
    "Resource": "arn:aws:ec2:*:*:network-interface/*",
    "Condition": {
     "StringEquals": {
      "ec2:CreateAction": "CreateNetworkInterface"
     },
     "ForAnyValue:StringEquals": {
      "aws:TagKeys": "ManagedByCloudShell",
      "aws:CalledVia": "cloudshell.amazonaws.com"
     }
    }
   },
   {
    "Sid": "AllowCreateNetworkInterfaceWithSubnetsAndSGViaCloudShell",
    "Effect": "Allow",
    "Action": [
     "ec2:CreateNetworkInterface"
```

```
    ],
    "Resource": [
     "arn:aws:ec2:*:*:subnet/*",
     "arn:aws:ec2:*:*:security-group/*"
    ],
    "Condition": {
     "ForAnyValue:StringEquals": {
      "aws:CalledVia": "cloudshell.amazonaws.com"
     }
    }
   },
   {
    "Sid": "AllowCreateNetworkInterfaceWithCloudShellTagViaCloudShell",
    "Effect": "Allow",
    "Action": [
     "ec2:CreateNetworkInterface"
    ],
    "Resource": "arn:aws:ec2:*:*:network-interface/*",
    "Condition": {
     "ForAnyValue:StringEquals": {
      "aws:TagKeys": "ManagedByCloudShell",
      "aws:CalledVia": "cloudshell.amazonaws.com"
     }
    }
   },
   {
    "Sid": "AllowCreateNetworkInterfacePermissionWithCloudShellTagViaCloudShell",
    "Effect": "Allow",
    "Action": [
     "ec2:CreateNetworkInterfacePermission"
    ],
    "Resource": "arn:aws:ec2:*:*:network-interface/*",
    "Condition": {
     "StringEquals": {
      "aws:ResourceTag/ManagedByCloudShell": ""
     },
     "ForAnyValue:StringEquals": {
      "aws:CalledVia": "cloudshell.amazonaws.com"
     }
    }
   },
   {
    "Sid": "AllowDeleteNetworkInterfaceWithCloudShellTagViaCloudShell",
    "Effect": "Allow",
```

```
    "Action": [
     "ec2:DeleteNetworkInterface"
    ],
    "Resource": "arn:aws:ec2:*:*:network-interface/*",
    "Condition": {
     "StringEquals": {
      "aws:ResourceTag/ManagedByCloudShell": ""
     },
     "ForAnyValue:StringEquals": {
      "aws:CalledVia": "cloudshell.amazonaws.com"
     }
    }
   }
  ]
}
```

Using IAM condition keys for VPC environments

You can use CloudShell-specific condition keys for VPC settings to provide additional permission controls for your VPC environments. You can also specify the subnets and security groups that the VPC environment can and can't use.

CloudShell supports the following condition keys in IAM policies:

- `CloudShell:VpcIds` – Allow or deny one or more VPCs
- `CloudShell:SubnetIds` – Allow or deny one or more subnets
- `CloudShell:SecurityGroupIds` – Allow or deny one or more security groups

> **Note**
>
> If the permissions for users with access to public CloudShell environments are modified to add restriction to the `cloudshell:createEnvironment` action, they can still access their existing public environment. However, if you want to modify an IAM policy with this restriction and disable their access to the existing public environment, you must first update the IAM policy with the restriction, and then ensure that every CloudShell user in your account manually deletes the existing public environment using the CloudShell web user interface (**Actions** → **Delete CloudShell environment**).

Example policies with condition keys for VPC settings

The following examples demonstrate how to use condition keys for VPC settings. After you create a policy statement with the desired restrictions, append the policy statement for the target user or role.

Ensure that users create only VPC environments and deny creation of public environments

To ensure that users can create only VPC environments, use the deny permission as shown in the following example:

```
{
  "Statement": [
    {
      "Sid": "DenyCloudShellNonVpcEnvironments",
      "Action": [
        "cloudshell:CreateEnvironment"
      ],
      "Effect": "Deny",
      "Resource": "*",
      "Condition": {
        "Null": {
          "cloudshell:VpcIds": "true"
        }
      }
    }
  ]
}
```

Deny users access to specific VPCs, subnets, or security groups

To deny users access to specific VPCs, use `StringEquals` to check the value of the `cloudshell:VpcIds` condition. The following example denies users access to `vpc-1` and `vpc-2`:

To deny users access to specific VPCs, use `StringEquals` to check the value of the `cloudshell:SubnetIds` condition. The following example denies users access to `subnet-1` and `subnet-2`:

To deny users access to specific VPCs, use `StringEquals` to check the value of the `cloudshell:SecurityGroupIds` condition. The following example denies users access to `sg-1` and `sg-2`:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "EnforceOutOfSecurityGroups",
      "Action": [
        "cloudshell:CreateEnvironment"
      ],
      "Effect": "Deny",
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "cloudshell:SecurityGroupIds": [
            "sg-1",
            "sg-2"
          ]
        }
      }
    }
  ]
}
```

Allow users to create environments with specific VPC configurations

To allow users access to specific VPCs, use `StringEquals` to check the value of the `cloudshell:VpcIds` condition. The following example allows users access to `vpc-1` and `vpc-2`:

To allow users access to specific VPCs, use `StringEquals` to check the value of the `cloudshell:SubnetIds` condition. The following example allows users access to `subnet-1` and `subnet-2`:

JSON

```
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "EnforceStayInSpecificSubnets",
      "Action": [
        "cloudshell:CreateEnvironment"
      ],
      "Effect": "Allow",
      "Resource": "*",
      "Condition": {
        "ForAllValues:StringEquals": {
          "cloudshell:SubnetIds": [
            "subnet-1",
            "subnet-2"
          ]
        }
      }
    }
  ]
}
```

To allow users access to specific VPCs, use `StringEquals` to check the value of the `cloudshell:SecurityGroupIds` condition. The following example allows users access to `sg-1` and `sg-2`:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "EnforceStayInSpecificSecurityGroup",
      "Action": [
        "cloudshell:CreateEnvironment"
      ],
      "Effect": "Allow",
      "Resource": "*",
      "Condition": {
        "ForAllValues:StringEquals": {
          "cloudshell:SecurityGroupIds": [
            "sg-1",
```

```
            "sg-2"
          ]
        }
      }
    }
  ]
}
```

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
- Identity and Access Management for AWS CloudShell
- Logging and monitoring in AWS CloudShell
- Compliance validation for AWS CloudShell
- Resilience in AWS CloudShell
- Infrastructure security in AWS CloudShell
- Security best practices for AWS CloudShell
- AWS CloudShell Security FAQs

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. You are also responsible for 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. For information about using CloudTrail trails to capture AWS activities, see Working with CloudTrail trails in the *AWS CloudTrail User Guide*.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- 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 140-3 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) 140-3.

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 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 AWS KMS keys—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 for AWS CloudShell

AWS Identity and Access Management (IAM) is an 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 CloudShell resources. IAM is an AWS service that you can use with no additional charge.

Topics

Audience

How you use AWS Identity and Access Management (IAM) differs based on your role:

- **Service user** - request permissions from your administrator if you cannot access features (see Troubleshooting AWS CloudShell identity and access)
- **Service administrator** - determine user access and submit permission requests (see How AWS CloudShell works with IAM)
- **IAM administrator** - write policies to manage access (see Identity-based policy examples for AWS CloudShell)

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be authenticated as the AWS account root user, an IAM user, or by assuming an IAM role.

You can sign in as a federated identity using credentials from an identity source like AWS IAM Identity Center (IAM Identity Center), single sign-on authentication, or Google/Facebook credentials. For more information about signing in, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

For programmatic access, AWS provides an SDK and CLI to cryptographically sign requests. For more information, see AWS Signature Version 4 for API requests in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity called the AWS account *root user* that has complete access to all AWS services and resources. We strongly recommend that you

don't use the root user for everyday tasks. For tasks that require root user credentials, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

As a best practice, require human users to use federation with an identity provider to access AWS services using temporary credentials.

A *federated identity* is a user from your enterprise directory, web identity provider, or Directory Service that accesses AWS services using credentials from an identity source. Federated identities assume roles that provide temporary credentials.

For centralized access management, we recommend AWS IAM Identity Center. For more information, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An *IAM user* is an identity with specific permissions for a single person or application. We recommend using temporary credentials instead of IAM users with long-term credentials. For more information, see Require human users to use federation with an identity provider to access AWS using temporary credentials in the *IAM User Guide*.

An *IAM group* specifies a collection of IAM users and makes permissions easier to manage for large sets of users. For more information, see Use cases for IAM users in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity with specific permissions that provides temporary credentials. You can assume a role by switching from a user to an IAM role (console) or by calling an AWS CLI or AWS API operation. For more information, see Methods to assume a role in the *IAM User Guide*.

IAM roles are useful for federated user access, temporary IAM user permissions, cross-account access, cross-service access, and applications running on Amazon EC2. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy defines permissions when associated with an identity or resource. AWS evaluates these policies when a principal makes a request. Most policies are stored in AWS as JSON documents. For

more information about JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Using policies, administrators specify who has access to what by defining which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. An IAM administrator creates IAM policies and adds them to roles, which users can then assume. IAM policies define permissions regardless of the method used to perform the operation.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you attach to an identity (user, group, or role). These policies control what actions identities can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

Identity-based policies can be *inline policies* (embedded directly into a single identity) or *managed policies* (standalone policies attached to multiple identities). To learn how to choose between managed and inline policies, see Choose between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples include IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. You must specify a principal in a resource-based policy.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Other policy types

AWS supports additional policy types that can set the maximum permissions granted by more common policy types:

- **Permissions boundaries** – Set the maximum permissions that an identity-based policy can grant to an IAM entity. For more information, see Permissions boundaries for IAM entities in the *IAM User Guide*.

- **Service control policies (SCPs)** – Specify the maximum permissions for an organization or organizational unit in AWS Organizations. For more information, see Service control policies in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – Set the maximum available permissions for resources in your accounts. For more information, see Resource control policies (RCPs) in the *AWS Organizations User Guide*.
- **Session policies** – Advanced policies passed as a parameter when creating a temporary session for a role or federated user. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How AWS CloudShell works with IAM

Before you use IAM to manage access to CloudShell, learn what IAM features are available to use with CloudShell.

IAM features you can use with AWS CloudShell

IAM feature	CloudShell support
Identity-based policies	Yes
Resource-based policies	No
Policy actions	Yes
Policy resources	Yes
Policy condition keys (service-specific)	Yes
ACLs	No
ABAC (tags in policies)	No

IAM feature	CloudShell support
Temporary credentials	Yes
Forward access sessions (FAS)	No
Service roles	No
Service-linked roles	No

To get a high-level view of how CloudShell and other AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

Identity-based policies for CloudShell

Supports identity-based policies: Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Identity-based policy examples for CloudShell

To view examples of CloudShell identity-based policies, see Identity-based policy examples for AWS CloudShell.

Resource-based policies within CloudShell

Supports resource-based policies: No

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified

principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Policy actions for CloudShell

Supports policy actions: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Include actions in a policy to grant permissions to perform the associated operation.

To see a list of CloudShell actions, see Actions defined by AWS CloudShell in the *Service Authorization Reference*. Some actions may have more than one API.

Policy actions in CloudShell use the following prefix before the action:

```
cloudshell
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [
      "cloudshell:action1",
      "cloudshell:action2"
         ]
```

To view examples of CloudShell identity-based policies, see Identity-based policy examples for AWS CloudShell.

Policy resources for CloudShell

Supports policy resources: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. As a best practice, specify a resource using its Amazon Resource Name (ARN). For actions that don't support resource-level permissions, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

To see a list of CloudShell resource types and their ARNs, see Resources defined by AWS CloudShell in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see Actions defined by AWS CloudShell.

To view examples of CloudShell identity-based policies, see Identity-based policy examples for AWS CloudShell.

Policy condition keys for CloudShell

Supports service-specific policy condition keys: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element specifies when statements execute based on defined criteria. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

To see a list of CloudShell condition keys, see Condition keys for AWS CloudShell in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see Actions defined by AWS CloudShell.

To view examples of CloudShell identity-based policies, see Identity-based policy examples for AWS CloudShell.

ACLs in CloudShell

Supports ACLs: No

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

ABAC with CloudShell

Supports ABAC (tags in policies): No

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes called tags. You can attach tags to IAM entities and AWS resources, then design ABAC policies to allow operations when the principal's tag matches the tag on the resource.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see Define permissions with ABAC authorization in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

Using temporary credentials with CloudShell

Supports temporary credentials: Yes

Temporary credentials provide short-term access to AWS resources and are automatically created when you use federation or switch roles. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM and AWS services that work with IAM in the *IAM User Guide*.

When you switch roles you will be using a different environment. You can't switch roles within the same AWS CloudShell environment.

Forward access sessions for CloudShell

Supports forward access sessions (FAS): No

Forward access sessions (FAS) use the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. For policy details when making FAS requests, see Forward access sessions.

Service roles for CloudShell

Supports service roles: No

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 Create a role to delegate permissions to an AWS service in the *IAM User Guide*.

> **Warning**
>
> Changing the permissions for a service role might break CloudShell functionality. Edit service roles only when CloudShell provides guidance to do so.

Service-linked roles for CloudShell

Supports service-linked roles: No

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

Identity-based policy examples for AWS CloudShell

By default, users and roles don't have permission to create or modify CloudShell resources. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Create IAM policies (console) in the *IAM User Guide*.

For details about actions and resource types defined by CloudShell, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for AWS CloudShell in the *Service Authorization Reference*.

Topics

Policy best practices

Identity-based policies determine whether someone can create, access, or delete CloudShell resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see Validate policies with IAM Access Analyzer in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API

operations are called, add MFA conditions to your policies. For more information, see Secure API access with MFA in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Using the CloudShell console

To access the AWS CloudShell console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the CloudShell resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (users or roles) with that policy.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that they're trying to perform.

To ensure that users and roles can still use the CloudShell console, also attach the CloudShell *ConsoleAccess* or *ReadOnly* AWS managed policy to the entities. For more information, see Adding permissions to a user in the *IAM User Guide*.

Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
```

```
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Troubleshooting AWS CloudShell identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with CloudShell and IAM.

Topics

- I am not authorized to perform an action in CloudShell
- I am not authorized to perform iam:PassRole
- I want to allow people outside of my AWS account to access my CloudShell resources

I am not authorized to perform an action in CloudShell

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *`my-example-widget`* resource but doesn't have the fictional `awes:`*`GetWidget`* permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 awes:GetWidget on resource: my-example-widget
```

In this case, the policy for the `mateojackson` user must be updated to allow access to the *my-example-widget* resource by using the `awes:GetWidget` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to CloudShell.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in CloudShell. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my AWS account to access my CloudShell resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether CloudShell supports these features, see How AWS CloudShell works with IAM.
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see Cross account resource access in IAM 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, 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. For more information on this policy, see AWSCloudShellFullAccess in the *AWS Managed Policy User Guide*.

> **Note**
>
> IAM identities with the following AWS managed policies can also launch CloudShell. However, these policies provide extensive permissions. So, we recommend that you only grant these policies if they're essential for an IAM user's job role.
>
> - Administrator: Provides IAM users with full access and allows them to delegate permissions to every service and resource in AWS.

- Developer power user: Enables IAM users to perform application development tasks and create and configure resources and services that support AWS aware application development.

For more information about attaching managed policies, see Adding IAM identity permissions (console) in the *IAM User Guide*.

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 CloudShellPolicy 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 IAM users to access CloudShell, but prevent them from forwarding the CloudShell environment credentials that are used to log in to AWS Management Console.

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

Name	Description of permission granted	Required to launch CloudShell?
`cloudshell:CreateEnvironment`	Creates a CloudShell environment, retrieves the layout at the start of the CloudShell session, and saves the current layout from the web application in the backend. This permission only expects * as the value for `Resource` as outlined in the section called "Examples of IAM policies for CloudShell".	Yes
`cloudshell:CreateSession`	Connects to a CloudShell environment from the AWS Management Console.	Yes
`cloudshell:GetEnvironmentStatus`	Read the status of a CloudShell environment.	Yes
`cloudshell:DeleteEnvironment`	Deletes a CloudShell environment.	No
`cloudshell:GetFileDownloadUrls`	Generates pre-signed Amazon S3 URLs that are used to download files through CloudShell using the CloudShell web interface. This is not available for VPC environments.	No

Name	Description of permission granted	Required to launch CloudShell?
`cloudshell:GetFileUploadUrls`	Generates pre-signed Amazon S3 URLs that are used to upload files through CloudShell using the CloudShell web interface. This is not available for VPC environments.	No
`cloudshell:DescribeEnvironments`	Describes the environments.	No
`cloudshell:PutCredentials`	Forwards the credentials used to log in to the AWS Management Console to CloudShell.	No
`cloudshell:StartEnvironment`	Starts a CloudShell environment that is stopped.	Yes
`cloudshell:StopEnvironment`	Stops a CloudShell environment that is running.	No
`cloudshell:ApproveCommand`	Approves a command sent to CloudShell from other AWS Service consoles.	No

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.

JSON

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

```
        "Resource": "*"
    }]
}
```

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": "AllowUsingCloudshell",
        "Effect": "Allow",
        "Action": [
            "cloudshell:*"
        ],
        "Resource": "*"
    },
    {
        "Sid": "DenyCredentialForwarding",
        "Effect": "Deny",
        "Action": [
            "cloudshell:PutCredentials"
        ],
        "Resource": "*"
    }]
}
```

The following policy allows IAM users to create AWS CloudShell environments.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
```

```
        "Sid": "CloudShellUser",
        "Effect": "Allow",
        "Action": [
            "cloudshell:CreateEnvironment",
            "cloudshell:CreateSession",
            "cloudshell:GetEnvironmentStatus",
            "cloudshell:StartEnvironment"
        ],
        "Resource": "*"
    }]
}
```

Required IAM permissions for creating and using CloudShell VPC environments

To create and use CloudShell VPC environments, the IAM administrator must enable access to VPC specific Amazon EC2 permissions. This section lists the Amazon EC2 permissions needed to create and use VPC environments.

To create VPC environments, the IAM policy assigned to your role must include the following Amazon EC2 permissions:

- `ec2:DescribeVpcs`
- `ec2:DescribeSubnets`
- `ec2:DescribeSecurityGroups`
- `ec2:DescribeDhcpOptions`
- `ec2:DescribeNetworkInterfaces`

- `ec2:CreateTags`
- `ec2:CreateNetworkInterface`
- `ec2:CreateNetworkInterfacePermission`

We recommend also including:

- **ec2:DeleteNetworkInterface**

Note

This permission is not mandatory, but this is required for CloudShell to clean up the ENI resource (ENIs created for CloudShell VPC environments are tagged with **ManagedByCloudShell** key) created by it. If this permission not in enabled, you must manually clean up the ENI resource after every CloudShell VPC environment use.

IAM policy granting full CloudShell access including access to VPC

The following example displays how to enable full permissions, including access to VPC, to CloudShell:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCloudShellOperations",
      "Effect": "Allow",
      "Action": [
        "cloudshell:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowDescribeVPC",
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeVpcs"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowCreateTagWithCloudShellKey",
      "Effect": "Allow",
```

```
      "Action": [
        "ec2:CreateTags"
      ],
      "Resource": "arn:aws:ec2:*:*:network-interface/*",
      "Condition": {
        "StringEquals": {
          "ec2:CreateAction": "CreateNetworkInterface"
        },
        "ForAnyValue:StringEquals": {
          "aws:TagKeys": "ManagedByCloudShell"
        }
      }
    },
    {
      "Sid": "AllowCreateNetworkInterfaceWithSubnetsAndSG",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:subnet/*",
        "arn:aws:ec2:*:*:security-group/*"
      ]
    },
    {
      "Sid": "AllowCreateNetworkInterfaceWithCloudShellTag",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface"
      ],
      "Resource": "arn:aws:ec2:*:*:network-interface/*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:TagKeys": "ManagedByCloudShell"
        }
      }
    },
    {
      "Sid": "AllowCreateNetworkInterfacePermissionWithCloudShellTag",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterfacePermission"
      ],
      "Resource": "arn:aws:ec2:*:*:network-interface/*",
```

```
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/ManagedByCloudShell": ""
        }
      }
    },
    {
      "Sid": "AllowDeleteNetworkInterfaceWithCloudShellTag",
      "Effect": "Allow",
      "Action": [
        "ec2:DeleteNetworkInterface"
      ],
      "Resource": "arn:aws:ec2:*:*:network-interface/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/ManagedByCloudShell": ""
        }
      }
    }
  ]
}
```

Using IAM condition keys for VPC environments

You can use CloudShell-specific condition keys for VPC settings to provide additional permission controls for your VPC environments. You can also specify the subnets and security groups that the VPC environment can and can't use.

CloudShell supports the following condition keys in IAM policies:

- `CloudShell:VpcIds` – Allow or deny one or more VPCs
- `CloudShell:SubnetIds` – Allow or deny one or more subnets
- `CloudShell:SecurityGroupIds` – Allow or deny one or more security groups

> **Note**
>
> If the permissions for users with access to public CloudShell environments are modified to add restriction to the `cloudshell:createEnvironment` action, they can still access their existing public environment. However, if you want to modify an IAM policy with this restriction and disable their access to the existing public environment, you must first

update the IAM policy with the restriction, and then ensure that every CloudShell user in your account manually deletes the existing public environment using the CloudShell web user interface (**Actions** → **Delete CloudShell environment**).

Example policies with condition keys for VPC settings

The following examples demonstrate how to use condition keys for VPC settings. After you create a policy statement with the desired restrictions, append the policy statement for the target user or role.

Ensure that users create only VPC environments and deny creation of public environments

To ensure that users can create only VPC environments, use the deny permission as shown in the following example:

```
{
  "Statement": [
    {
      "Sid": "DenyCloudShellNonVpcEnvironments",
      "Action": [
        "cloudshell:CreateEnvironment"
      ],
      "Effect": "Deny",
      "Resource": "*",
      "Condition": {
        "Null": {
          "cloudshell:VpcIds": "true"
        }
      }
    }
  ]
}
```

Deny users access to specific VPCs, subnets, or security groups

To deny users access to specific VPCs, use `StringEquals` to check the value of the `cloudshell:VpcIds` condition. The following example denies users access to `vpc-1` and `vpc-2`:

To deny users access to specific VPCs, use `StringEquals` to check the value of the `cloudshell:SubnetIds` condition. The following example denies users access to `subnet-1` and `subnet-2`:

To deny users access to specific VPCs, use `StringEquals` to check the value of the `cloudshell:SecurityGroupIds` condition. The following example denies users access to `sg-1` and `sg-2`:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "EnforceOutOfSecurityGroups",
      "Action": [
        "cloudshell:CreateEnvironment"
      ],
      "Effect": "Deny",
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "cloudshell:SecurityGroupIds": [
            "sg-1",
            "sg-2"
          ]
        }
      }
    }
  ]
}
```

Allow users to create environments with specific VPC configurations

To allow users access to specific VPCs, use `StringEquals` to check the value of the `cloudshell:VpcIds` condition. The following example allows users access to `vpc-1` and `vpc-2`:

To allow users access to specific VPCs, use `StringEquals` to check the value of the `cloudshell:SubnetIds` condition. The following example allows users access to `subnet-1` and `subnet-2`:

JSON

```
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "EnforceStayInSpecificSubnets",
      "Action": [
        "cloudshell:CreateEnvironment"
      ],
      "Effect": "Allow",
      "Resource": "*",
      "Condition": {
        "ForAllValues:StringEquals": {
          "cloudshell:SubnetIds": [
            "subnet-1",
            "subnet-2"
          ]
        }
      }
    }
  ]
}
```

To allow users access to specific VPCs, use `StringEquals` to check the value of the `cloudshell:SecurityGroupIds` condition. The following example allows users access to `sg-1` and `sg-2`:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "EnforceStayInSpecificSecurityGroup",
      "Action": [
        "cloudshell:CreateEnvironment"
      ],
      "Effect": "Allow",
      "Resource": "*",
      "Condition": {
        "ForAllValues:StringEquals": {
          "cloudshell:SecurityGroupIds": [
            "sg-1",
```

```
            "sg-2"
          ]
        }
      }
    }
  ]
}
```

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*.

Permissions for accessing Amazon Q CLI features in CloudShell

To use Amazon Q CLI features in CloudShell, such as inline suggestions, chat, and translate, make sure you have the required IAM permissions. If you're unable to access Amazon Q CLI features in CloudShell, contact your administrator to provide you with the necessary IAM permissions. For more information, see Identity-based policy examples for Amazon Q Developer in the *Amazon Q Developer 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

The following table lists the AWS CloudShell events that are saved in the CloudTrail log file.

Note

AWS CloudShell event that includes:

- * indicates that it is a non-mutating (read-only) API call.
- The word `Environment` relates to the lifecycle of the compute environment that hosts the shell experience.
- The word `Layout` restores all the browser tabs in the CloudShell terminal.

CloudShell Events in CloudTrail

Event name	Description
`createEnvironment`	Occurs when a CloudShell environment is created.
`createSession`	Occurs when a CloudShell environment is connected from the AWS Management Console.
`deleteEnvironment`	Occurs when a CloudShell environment is deleted.
`deleteSession`	Occurs when the session in the CloudShell tab that is running in the current browser tab is deleted.
`getEnvironmentStatus*`	Occurs when the status of a CloudShell environment is retrieved.
`getFileDownloadUrls*`	Occurs when pre-signed Amazon S3 URLs that are used to download files through CloudShel l using the CloudShell web interface are generated.
`getFileUploadUrls*`	Occurs when pre-signed Amazon S3 URLs that are used to upload files through CloudShel l using the CloudShell web interface are generated.
`cloudshell:DescribeEnvironments`	Describes the environments.
`getLayout*`	Occurs when the CloudShell layout at the start of the session is retrieved.
`putCredentials`	Occurs when the credentials used to log in to the AWS Management Console to CloudShell are forwarded.

Event name	Description
redeemCode*	Occurs when the workflow to retrieve refresh token in the CloudShell environment begins. You can later use this token in the putCredentials command to access the CloudShell environment.
sendHeartBeat	Occurs to confirm that the CloudShell session is active.
startEnvironment	Occurs when a CloudShell environment is started.
stopEnvironment	Occurs when a running CloudShell environment is stopped.
updateLayout	Occurs when the current layout from the web application in the backend is saved.

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

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.

Service	SDK	SOC 1,2,3
AWS CloudShell	CloudShell	✓

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.

Service	SDK	PCI
AWS CloudShell	CloudShell	✓

ISO and CSA STAR Certifications and Services

AWS has certification for compliance with ISO/IEC 27001:2013, 27017:2015, 27018:2019, 27701:2019, 22301:2019, 9001:2015, and CSA STAR CCM v4.0.

Service	SDK	ISO and CSA STAR Certifications and Services
AWS CloudShell	CloudShell	✓

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.

Service	SDK	FedRAMP Moderate (East/West)	FedRAMP High (GovCloud)
AWS CloudShell	CloudShell	✓	✓

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.

Service	SDK	DoD CC SRG IL2 (East/West)	DoD CC SRG IL2 (GovCloud)	DoD CC SRG IL4 (GovCloud)	DoD CC SRG IL5 (GovCloud)	DoD CC SRG IL6 (AWS Secret Region)
AWS CloudShell	CloudShell	✓	✓	✓	✓	N/A

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.

Service	SDK	HIPAA BAA
AWS CloudShell	CloudShell	✓

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 addressing the requirements of the Australian Government Information Security Manual (ISM) produced by the Australian Cyber Security Centre (ACSC).

Service	Namespace*	IRAP protected
AWS CloudShell	N/A	✓

*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.

Service	SDK	US-East(Ohio)	US-East(N.Virginia)	US-West(Oregon)	US-West(N.California)	Singapore	Seoul
AWS CloudShell	CloudShell	✓	✓	✓	N/A	N/A	N/A

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".

Service	SDK	C5
AWS CloudShell	CloudShell	✓

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.

Service	SDK	ENS High
AWS CloudShell	CloudShell	✓

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.

Service	SDK	FINMA
AWS CloudShell	CloudShell	✓

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.

Service	SDK	PiTuKri
AWS CloudShell	CloudShell	✓

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 CSPM – 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 the following feature to support your data resiliency and backup needs:

- 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 with AWS CloudShell.

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.

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

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, we recommend that you 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 (for example, downloading and uploading files) required by their role. For more information, see Managing AWS CloudShell access and usage with IAM policies.
- 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 about using safe paste for multiline text, see Using Safe Paste for multiline text.
- Be familiar with the Shared Security Responsibility Model if you install 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 about modifying the default shell environment, see Modifying your shell with scripts.
- Store your code securely in a version control system.

AWS CloudShell Security FAQs

The following are answers to frequently asked questions about security for CloudShell.

What AWS processes and technologies 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.

Back to list of security FAQs

Is it possible to restrict network access to CloudShell?

For public environments, it is not possible to restrict network access. If you want to restrict network access, you must enable permission to create only VPC environments and deny creation of public environments.

For more information, see Ensure that users create only VPC environments and deny creation of public environments.

For CloudShell VPC environments, network settings are inherited from your VPC. Using CloudShell in a VPC enables you to control your CloudShell VPC environment's network access.

Back to list of security FAQs

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

Where is my $HOME directory actually stored in the AWS Cloud?

For Public environments, the infrastructure for storing data in your $HOME is provided by Amazon S3.

For VPC environments, your $HOME directory is deleted when your VPC environment times out (after 20-30 minutes of inactivity), or when you delete or restart your environment.

Back to list of security FAQs

Is it possible to encrypt my $HOME directory?

No, it is not possible to encrypt your $HOME directory with your own key. But CloudShell encrypts your $HOME directory content while storing it in Amazon S3.

Back to list of security FAQs

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

Can I restrict data ingress or egress for my CloudShell?

To restrict ingress or egress, we recommend that you use a CloudShell VPC environment. The $HOME directory of a VPC environment is deleted when your VPC environment times out (after 20-30 minutes of inactivity), or when you delete or restart your environment. In the **Actions** menu, the upload and download options are not available for VPC environments.

Back to list of security FAQs

AWS CloudShell compute environment: specifications and software

When you launch AWS CloudShell, a compute environment that's based on Amazon Linux 2023 is created to host the shell experience. The environment is configured with compute resources (vCPU and memory) and provides a wide range of pre-installed software that can be accessed from the command line interface. Make sure that any software you install in the compute environment is patched and up to date. 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.

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

Name	Description	Version information
Bash	The Bash shell is the default shell application for AWS CloudShell.	`bash --version`
PowerShell (pwsh)	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.	`pwsh --version`
Z Shell (zsh)	The Z Shell, also known as `zsh`, is an extended version of the Bourne Shell that offers enhanced customization support for themes and plugins.	`zsh --version`

AWS command line interfaces (CLI)

CLI

Name	Description	Version information
AWS CDK Toolkit CLI	The AWS CDK Toolkit, the CLI command, `cdk`, is the primary tool that interacts with your AWS CDK app. It executes your app, interroga	`cdk --version`

Name	Description	Version information
	tes the application model you defined, and produces and deploys the AWS CloudForm ation templates generated by the AWS CDK. For more information, see AWS CDK Toolkit.	
AWS CLI	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 informati on, see Manage AWS services from CLI in CloudShell. For information about how you can ensure that you're using the most up-to-date version AWS CLI version 2, see Installing AWS CLI to your home directory.	`aws --version`

Name	Description	Version information
EB CLI	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 Using the Elastic Beanstalk command line interface (EB CLI) in the *AWS Elastic Beanstalk Developer Guide*.	`eb --version`
Amazon ECS CLI	Amazon Elastic Container Service (Amazon ECS) command line interface (CLI) provides high-leve l commands to simplify creating, updating, and monitoring clusters and tasks. For more information, see Using the Amazon ECS Command Line Interface in the *Amazon Elastic Container Service Developer Guide*.	`ecs-cli --version`

Name	Description	Version information
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. 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 AWS SAM CLI command reference in the *AWS Serverless Application Model Developer Guide*.	`sam --version`

Name	Description	Version information
AWS Tools for PowerShell	The AWS Tools for PowerShell are PowerShell modules that are built on the functionality exposed by the SDK for .NET. With AWS Tools for PowerShell, you can script operations on your AWS resources from the PowerShell command line. AWS CloudShell pre-installs the modularized version (AWS.Tools) of the AWS Tools for PowerShell. For more information, see Using the AWS Tools for PowerShell in the *AWS Tools for PowerShell User Guide*.	`pwsh --Command 'Get-AWSPowerShellVersion'`

Runtimes and AWS SDKs: Node.js and Python 3

Runtimes and AWS SDKs

Name	Description	Version information
Node.js (with npm)	Node.js is a JavaScript runtime that's designed to make it easier to apply asynchronous programming techniques. For more information, see the documentation on the official Node.js site.	• Node.js: `node --version` • npm: `npm --version`

Name	Description	Version information
	npm is a package manager that provides access to an online registry of JavaScript modules. For more informati on, see the documentation on the official npm site.	
SDK for JavaScript in Node.js	The software developme nt kit (SDK) helps simplify coding by providing JavaScrip t objects for AWS services including Amazon S3, Amazon EC2, DynamoDB, and Amazon SWF. For more information, see the AWS SDK for JavaScript Developer Guide.	`npm -g ls --depth 0 2>/dev/null | grep aws-sdk`

Name	Description	Version information
Python	Python 3 is 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. 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.	• Python 3: `python3 --version` • pip: `pip3 --version`

Name	Description	Version information
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.	`pip3 list | grep boto3`

Development tools and shell utilities

Development tools and shell utilities

Name	Description	Version information
bash-completion	bash-completion is a collection of shell functions that allow the autocompletion of partially typed commands or arguments by pressing the **Tab** key. You can find the packages that bash-completion supports in `/usr/share/bash-completion/completions` . To set up autocomplete for a package's commands, the program file must be sourced. For example, to	`dnf info bash-completion`

Name	Description	Version information
	set up autocomplete for Git commands, add the following line to `.bashrc` so the feature is available whenever your AWS CloudShell session starts: `source /usr/share/bash-completion/completions/git` If you want to use custom completion scripts, add them to your persistent home directory (`$HOME`) and source them directly in `.bashrc`. For more information, see the project's README page on GitHub.	
cqlsh-expansion	`cqlsh-expansion` is a toolkit that includes cqlsh and helpers that are preconfigured for Amazon Keyspaces while maintaining full compatibility with Apache Cassandra. For more information, see Using cqlsh to connect to Amazon Keyspaces in *Amazon Keyspaces (for Apache Cassandra) Developer Guide*.	`cqlsh-expansion --version`

Name	Description	Version information
Docker	Docker is an open platform for developing, shipping and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. It enables you to build Dockerfiles inside AWS CloudShell, and build Docker assets with CDK. For information about which AWS Regions are supported with Docker, see Supported AWS Regions for AWS CloudShell. You should be aware that Docker has limited space in the environment. If you have large individual images, or too many pre-existing Docker images, it can cause issues. For more information on Docker, see the Docker Documentation guide.	`docker --version`
Git	Git is a distributed version control system that supports modern software development practices through branch workflows and content staging. For more information, see the documentation page on Git's official site.	`git --version`

Name	Description	Version information
iputils	The iputils package contains utilities for Linux networking. For more information about the utilities provided, see the iputils repository on GitHub.	Examples for an iputils tool: `arping -V`
jq	The jq utility parses JSON-formatted data to produce output that's modified by command line filters. For more information, see the jq manual hosted on GitHub.	`jq --version`
kubectl	kubectl is a command line tool for communicating with a Kubernetes cluster's control plane, using the Kubernetes API.	`kubectl --version`
make	The make utility uses `makefiles` to automate sets of tasks and organize code compilation. For more information, see the GNU Make documentation.	`make --version`
man	The man command provides manual pages for command line utilities and tools. For example, `man ls` returns the manual page for the `ls` command that lists the contents of directories. For more information, see the Wikipedia entry on man page.	`man --version`

Name	Description	Version information
nano	nano is a small and user-friendly editor for text-based interface. For more information, see the GNU nano documentation.	`nano --version`
OpenJDK 21	Amazon Corretto 21 is a Long-Term Supported (LTS) distribution of OpenJDK 21. Amazon Corretto is a no-cost, multiplatform, productio n-ready distribution of the Open Java Development Kit (OpenJDK). For more information, see What is Amazon Corretto 21? in the *Corretto 21 User Guide*.	`java -version`
procps	procps is a system administr ation utility that you can use to monitor and halt currently running processes. For more information, see the README file that lists programs that can be run with procps.	`ps --version`
psql	PostgreSQL is a powerful, open source database system that uses standard SQL capabilities while providing robust features for securely managing and scaling complex data operations. For more information, see What is PostgreSQL.	`psql --version`

Name	Description	Version information
SSH client	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 informati on, see the OpenSSH site maintained by the OpenBSD.	`ssh -V`
sudo	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 Sudo Manual.	`sudo --version`
tar	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 GNU tar documentation.	`tar --version`
tmux	tmux is a terminal multiplex er that you can use to run different programs simultane ous in multiple windows. For more information, see a blog that provides a concise introduction to tmux.	`tmux -V`

Name	Description	Version information
vim	vim is a customizable editor that you can interact with through a text-based interface. For more informati on, see the documenta tion resources provided on vim.org.	`vim --version`
wget	wget is a computer program used to retrieve content from web servers specified by endpoints in the command line. For more information, see the GNU Wget documenta tion.	`wget --version`

zip/unzip	The zip/unzip utilities use an archive file format that delivers lossless data compression without data loss. Call the zip command to group and compress files in a single archive. Use unzip to extract files from an archive into a specified directory.	`unzip --version`
		
		`zip --version`

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.

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:

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

2. Unzip the zipped folder:

```
unzip awscliv2.zip
```

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

```
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:

```
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:

```
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 DNF package-management utility to install `cowsay`, which generates ASCII art pictures of a cow with a message:

```
sudo dnf install cowsay
```

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

> **Important**
>
> Package manage utilities such as dnf 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.

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:

   ```
   :wq
   ```

4. Use the `source` command to reload the `.bashrc` file:

   ```
   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 AL2 to AL2023

AWS CloudShell, which was based on Amazon Linux 2 (AL2), has migrated to Amazon Linux 2023 (AL2023). For more information about AL2023, see What is Amazon Linux 2023 (AL2023) in the *Amazon Linux 2023 User Guide*.

With 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.

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 is no longer shipped with your CloudShell environment.

For more information about the key differences between AL2 and AL2023, see Comparing Amazon Linux 2 and Amazon Linux 2023 in the *Amazon Linux 2023 User Guide*.

If you've any questions, contact Support. You can also search for answers and post questions in AWS re:Post. 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 the migration to AL2023 affect any of my other AWS resources, such as Amazon EC2 instances running on AL2?
- What are the packages that will be changed with the migration to AL2023?
- Can I opt-out from migration?
- Can I create a backup of my AWS CloudShell environment?

Will the migration to AL2023 affect any of my other AWS resources, such as 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 might 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 have been 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. AWS CloudShell will continue delivering these packages, with the 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?

No, you can't opt-out from migration. AWS CloudShell environments are managed by AWS, therefore, all the environments have been 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

Denied access

Issue: When you try to launch CloudShell from the AWS Management Console, you get the message "Unable to start the environment. To retry, refresh the browser or restart by selecting Actions, Restart AWS CloudShell ". 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.

(back to top)

Insufficient permissions

Issue: When you try to launch CloudShell from the AWS Management Console, you get the message "Unable to start the environment. You don't have required permissions. Ask your IAM administrator to grant access to AWS CloudShell". You're denied access and notified that you don't have required permissions.

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.

(back to top)

Unable to access AWS CloudShell command line

Issue: After modifying a file that the compute environment uses, you can't access the command line in AWS CloudShell.

Solution: If you lose access after incorrectly modifying `.bashrc` or any other file, you can return AWS CloudShell to its default settings by deleting your home directory.

(back to top)

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.

Solution: Due to the fact that ICMP is not supported in AWS CloudShell, you can run the following command to install Netcat. Netcat is a is a computer networking utility for reading from, and writing to, network connections using TCP or UDP.

```
sudo yum install nc
nc -zv www.amazon.com 443
```

(back to top)

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.

(back to top)

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)

Unsupported Web Sockets cause a failure to start CloudShell sessions

Issue: When you try 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 allowlisting 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.
> 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 you update your URL allowlist. For more information, see the StartSession operation in the *AWS Systems Manager API Reference*.

(back to top)

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.

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

```
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)

Docker is not running when using AWS CloudShell

Issue: Docker is not running properly when using AWS CloudShell. You receive the following error message: `docker: Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?`.

Solution: Try restarting your environment. This error message can occur when you run Docker in AWS CloudShell in a GovCloud Region. Make sure that you are running Docker in the supported AWS Regions. For a list of Regions in which Docker is available, see Supported AWS Regions for AWS CloudShell.

Docker has ran out of disk space

Issue: You are receiving the following error message: `ERROR: failed to solve: failed to register layer: write [...]: no space left on device`.

Cause: The Dockerfile is exceeding the available disk space in AWS CloudShell. This can be caused due to large individual images or too many pre-existing Docker images.

Solution: Run `df -h` to find the disk usage. Run `sudo du -sh /folder/folder1` to asses the size of certain folders that you feel may be large and consider deleting other files to free up space. One option would be to consider removing unused Docker images by running `docker rmi`. You should be aware that Docker has limited space in the environment, for more information on Docker, see the Docker Documentation guide.

`docker push` is timing out and keeps retrying

Issue: When you run `docker push` it is timing out and continues to retry with no success.

Cause: This can be caused as a result of missing permissions, pushing to the wrong repository or a lack of authentication.

Solution: To try and resolve this issue make sure you are pushing to the correct repository. Run `docker login` to properly authenticate. Make sure that you have all the required permissions for pushing to an Amazon ECR repository.

Unable to access resources within VPC from my AWS CloudShell VPC environment

Issue: Unable to access resources within the VPC while using my AWS CloudShell VPC environment.

Cause: Your AWS CloudShell VPC environment inherits the network settings of your VPC.

Solution: To resolve this issue make sure that your VPC is set up correctly to access your resources. For more information, see VPC documentation Connect your VPC to other networks and the and the Network Access Analyzer documentation Network Access Analyzer. You can find the IPv4 address that the AWS CloudShell VPC environment is using, by running the command **`ip -a`** inside your environment in the command line prompt, or on the VPC Console page.

The ENI used by AWS CloudShell for my VPC environment is not cleaned up

Issue: Unable to clean up the ENI used by AWS CloudShell for my VPC environment.

Cause: `ec2:DeleteNetworkInterface` permission is not enabled for your role.

Solution: To resolve this issue, make sure that `ec2:DeleteNetworkInterface` permission is enabled for your role as shown in the following sample script:

```
{
  "Effect": "Allow",
  "Action": [
    "ec2:DeleteNetworkInterface"
  ],
  "Condition": {
    "StringEquals": {
      "aws:ResourceTag/ManagedByCloudShell": ""
    }
  },
  "Resource": "arn:aws:ec2:*:*:network-interface/*"
}
```

User with `CreateEnvironment` permission for only VPC environments also has access to public AWS CloudShell environments

Issue: User restricted with `CreateEnvironment` permission for only VPC environments is also able to access public AWS CloudShell environments.

Cause: When you limit `CreateEnvironment` permissions for creation of VPC environments only and if you have already created a public environment, you will keep your access to the existing public CloudShell environment until this environment is deleted using the web user interface. But if you have never used CloudShell before, you will not have access to public environments.

Solution: To restrict access to public AWS CloudShell environments, the IAM administrator must first update the IAM policy with the restriction, and then the user must manually delete the existing public environment using the AWS CloudShell web user interface. (**Actions** → **Delete CloudShell environment**).

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 CloudShell, Docker, and CloudShell VPC environment:

- US East (Ohio)
- US East (N. Virginia)
- US West (N. California)
- US West (Oregon)
- Africa (Cape Town)
- Asia Pacific (Hong Kong)
- Asia Pacific (Jakarta)
- Asia Pacific (Mumbai)
- Asia Pacific (Osaka)
- Asia Pacific (Seoul)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)
- Asia Pacific (Tokyo)
- Canada (Central)
- Europe (Frankfurt)
- Europe (Ireland)
- Europe (London)
- Europe (Milan)
- Europe (Paris)
- Europe (Stockholm)
- Middle East (Bahrain)
- Middle East (UAE)
- South America (São Paulo)

GovCloud Regions

The following are the supported GovCloud Regions for CloudShell:

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

> **Note**
>
> Docker and CloudShell VPC environment are available in GovCloud Regions.

Service quotas and restrictions for AWS CloudShell

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

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.

> **Note**
>
> CloudShell VPC environments do not have persistent storage. The $HOME directory is deleted when your VPC environment times out (after 20-30 minutes of inactivity), or when you delete your environment.

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.

> **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

Each AWS Region in your AWS account has a monthly usage quota for AWS CloudShell. This quota combines the total time spent using CloudShell by all IAM principals in that Region. If you attempt to access CloudShell after you reached the monthly quota for that Region, a message displays to explain why the shell environment can't be started.

To request an increase using the Service Quotas console

You can request an increase for your monthly usage quotas by opening the Service Quotas console. For more information, see Requesting a quota increase in the *Service Quotas User Guide*.

Concurrent shells

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

To request an increase using the Service Quotas console

You can request an quota increase for each Region by opening the Service Quotas console. For more information, see Requesting a quota increase in the *Service Quotas User Guide*.

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.
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.

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.

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

VPC environments

You can only create up to two VPC environments per IAM principal.

Note

There is no charge to connect to your private VPC and access the resources within it.
Data transfers within your Private VPC is included in your VPC billing, and data transfers between your VPCs through CloudShell are charged at the same cost as your current CloudShell.

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.

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 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*.

Change	Description	Date
Amazon Q CLI in AWS CloudShell	Added support for using Amazon Q CLI features in AWS CloudShell.	October 2, 2024
Amazon VPC support for AWS CloudShell in certain Regions	Added support for creating and using AWS CloudShell VPC environments in certain Regions.	June 13, 2024
New tutorials have been added to the AWS CloudShell User Guide	Two new tutorials have been added that details how to build a Docker container inside AWS CloudShell and push it to an Amazon ECR repository, and how to deploy a Lambda function via AWS CDK.	December 27, 2023
Docker containers supported with AWS CloudShell in certain Regions	Support for Docker container s with AWS CloudShell has been added in certain Regions.	December 27, 2023
AWS CloudShell has migrated to now use Amazon Linux 2023 (AL2023)	AWS CloudShell now uses AL2023 and has migrated from Amazon Linux 2.	December 4, 2023
New AWS Regions for AWS CloudShell	AWS CloudShell is now generally available in the following AWS Regions:	June 16, 2023

	• 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)	
Launch AWS CloudShell on the Console Toolbar	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	AWS CloudShell is now available in the following AWS Regions: • Canada (Central) • Europe (London) • South America (São Paulo)	October 6, 2022
AWS CloudShell supported in US AWS GovCloud	AWS CloudShell is now supported in the AWS GovCloud (US) Region.	June 29, 2022
Security FAQs	Additional FAQs focused on security issues.	April 14, 2022

Web Sockets	Added section to network requirements that explains CloudShell's use of the WebSocket protocol.	March 21, 2022
Troubleshooting arrow keys in PowerShell	Follow the steps to fix arrow keys that incorrectly output letters when pressed.	February 7, 2022
Tab key autocomplete	New documentation that explains how to use bash-completion, which allows the autocompletion of partially typed commands or arguments by pressing the **Tab** key.	September 24, 2021
Specifying AWS Regions	Documentation on specifying default AWS Region for AWS CLI commands.	May 11, 2021
Formatting in PDF and Kindle versions	Fixed image sizes and text in table cells.	March 10, 2021
General availability (GA) release of AWS CloudShell in selected AWS Regions	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