Command completion

The AWS Command Line Interface (AWS CLI) includes a bash-compatible command-completion feature that enables you to use the **Tab** key to complete a partially entered command. On most systems you need to configure this manually.

For information on the AWS CLI version 2 auto-prompt feature instead, see Having the AWS CLI prompt you for commands (p. 133).

Topics

- How it works (p. 90)
- Configuring command completion on Linux or macOS (p. 90)
- Configuring command completion on Windows (p. 93)

How it works

When you partially enter a command, parameter, or option, the command-completion feature either automatically completes your command or displays a suggested list of commands. To prompt command completion, you partially enter in a command and press the completion key, which is typically *Tab* in most shells.

The following examples show different ways that you can use command completion:

- Partially enter a command and press *Tab* to display a suggested list of commands.

```
$ aws dynamodb dTAB
delete-backup                       describe-global-table
delete-item                         describe-global-table-settings
delete-table                        describe-limits
describe-backup                     describe-table
describe-continuous-backups         describe-table-replica-auto-scaling
describe-contributor-insights       describe-time-to-live
describe-endpoints
```

- Partially enter a parameter and press *Tab* to display a suggested list of parameters.

```
$ aws dynamodb delete-table --TAB
--ca-bundle              --endpoint-url           --profile
--cli-connect-timeout    --generate-cli-skeleton  --query
--cli-input-json         --no-paginate            --region
--cli-read-timeout       --no-sign-request        --table-name
--color                  --no-verify-ssl          --version
--debug                  --output
```

- Enter a parameter and press *Tab* to display a suggested list of resource values. This feature is available only in the AWS CLI version 2.

```
$ aws dynamodb db delete-table --table-name TAB
Table 1                  Table 2                  Table 3
```

Configuring command completion on Linux or macOS

To configure command completion on Linux or macOS, you must know the name of the shell you're using and the location of the `aws_completer` script.

Note
Command completion is automatically configured and enabled by default on Amazon EC2 instances that run Amazon Linux.

Topics

- Confirm the completer's folder is in your path (p. 91)
- Enable command completion (p. 92)
- Verify command completion (p. 93)

Confirm the completer's folder is in your path

For the AWS completer to work successfully, the `aws_completer` needs to be in your shell's path. The `which` command can check if the completer is in your path.

```
$ which aws_completer
/usr/local/bin/aws_completer
```

If the which command can't find the completer, then use the following steps to add the completer's folder to your path.

Step 1: Locate the AWS completer

The location of the AWS completer can vary depending on the installation method used.

- **Package Manager** - Programs such as `pip`, `yum`, `brew`, and `apt-get` typically install the AWS completer (or a symlink to it) to a standard path location.
  - If you used `pip` **without** the `--user` parameter, the default path is `/usr/local/bin/aws_completer`.
  - If you used `pip` **with** the `--user` parameter the default path is `/home/`*`username`*`/.local/bin/aws_completer`.
- **Bundled Installer** - If you used the bundled installer, the default path is `/usr/local/bin/aws_completer`.

If all else fails, you can use the `find` command to search your file system for the AWS completer.

```
$ find / -name aws_completer
/usr/local/bin/aws_completer
```

Step 2: Identify your shell

To identify which shell you're using, you can use one of the following commands.

- **echo $SHELL** – Displays the shell's program file name. This usually matches the name of the in-use shell, unless you launched a different shell after logging in.

  ```
  $ echo $SHELL
  /bin/bash
  ```

- **ps** – Displays the processes running for the current user. One of them is the shell.

  ```
  $ ps
    PID TTY          TIME CMD
  ```

```
 2148 pts/1    00:00:00 bash
 8756 pts/1    00:00:00 ps
```

Step 3: Add the completer to your path

1. Find your shell's profile script in your user folder.

```
$ ls -a ~/
.  ..  .bash_logout  .bash_profile  .bashrc  Desktop  Documents  Downloads
```

   - **Bash**– `.bash_profile`, `.profile`, or `.bash_login`
   - **Zsh**– `.zshrc`
   - **Tcsh**– `.tcshrc`, `.cshrc`, or `.login`
2. Add an export command at the end of your profile script that's similar to the following example. Replace */usr/local/bin/* with the folder that you discovered in the previous section.

```
export PATH=/usr/local/bin/:$PATH
```

3. Reload the profile into the current session to put those changes into effect. Replace *.bash_profile* with the name of the shell script you discovered in the first section.

```
$ source ~/.bash_profile
```

Enable command completion

After confirming the completer is in your path, enable command completion by running the appropriate command for the shell that you're using. You can add the command to your shell's profile to run it each time you open a new shell. In each command, replace the */usr/local/bin/* path with the one found on your system in Confirm the completer's folder is in your path (p. 91).

- **bash** – Use the built-in command `complete`.

```
$ complete -C '/usr/local/bin/aws_completer' aws
```

  Add the previous command to `~/.bashrc` to run it each time you open a new shell. Your `~/.bash_profile` should source `~/.bashrc` to ensure that the command is also run in login shells.
- **zsh** – To run command completion, you need to run `bashcompinit` by adding the following autoload line at the end of your `~/.zshrc` profile script.

```
$ autoload bashcompinit && bashcompinit
$ autoload -Uz compinit && compinit
```

  To enable command completion, use the built-in command `complete`.

```
$ complete -C '/usr/local/bin/aws_completer' aws
```

  Add the previous commands to `~/.zshrc` to run it each time you open a new shell.
- **tcsh** – Complete for `tcsh` takes a word type and pattern to define the completion behavior.

```
> complete aws 'p/*/`aws_completer`/'
```

Add the previous command to `~/.tschrc` to run it each time you open a new shell.

After you've enabled command completion, Verify command completion (p. 93) is working.

Verify command completion

After enabling command completion, reload your shell, enter a partial command, and press **Tab** to see the available commands.

```
$ aws sTAB
s3              ses             sqs             sts             swf
s3api           sns             storagegateway  support
```

Configuring command completion on Windows

Note
For information on how PowerShell handles their completion, including their various completion keys, see about_Tab_Expansion in the *Microsoft PowerShell Docs*.

To enable command completion for PowerShell on Windows, complete the following steps in PowerShell.

1. Open your `$PROFILE` with the following command.

   ```
   PS C:\> Notepad $PROFILE
   ```

   If you do not have a `$PROFILE`, create a user profile using the following command.

   ```
   PS C:\> if (!(Test-Path -Path $PROFILE ))
   { New-Item -Type File -Path $PROFILE -Force }
   ```

   For more information on PowerShell profiles, see How to Use Profiles in Windows PowerShell ISE on the *Microsoft Docs* website.
2. To enable command completion, add the following code block to your profile, save, and then close the file.

   ```
   Register-ArgumentCompleter -Native -CommandName aws -ScriptBlock {
       param($commandName, $wordToComplete, $cursorPosition)
           $env:COMP_LINE=$wordToComplete
           if ($env:COMP_LINE.Length -lt $cursorPosition){
               $env:COMP_LINE=$env:COMP_LINE + " "
           }
           $env:COMP_POINT=$cursorPosition
           aws_completer.exe | ForEach-Object {
               [System.Management.Automation.CompletionResult]::new($_, $_,
    'ParameterValue', $_)
           }
           Remove-Item Env:\COMP_LINE
           Remove-Item Env:\COMP_POINT
   }
   ```
3. After enabling command completion, reload your shell, enter a partial command, and press **Tab** to cycle through the available commands.

   ```
   $ aws sTab
   ```

```
$ aws s3
```

To see all available commands available to your completion, enter a partial command and press **Ctrl** + **Space**.

```
$ aws sCtrl + Space
s3              ses             sqs             sts             swf
s3api           sns             storagegateway  support
```

AWS CLI retries

This topic describes how the AWS CLI might see calls to AWS services fail due to unexpected issues. These issues can occur on the server side or might fail due to rate limiting from the AWS service you're attempting to call. These kinds of failures usually don't require special handling and the call is automatically made again, often after a brief waiting period. The AWS CLI provides many features to assist in retrying client calls to AWS services when these kinds of errors or exceptions are experienced.

Topics

- Available retry modes (p. 94)
- Configuring a retry mode (p. 96)
- Viewing logs of retry attempts (p. 97)

Available retry modes

The AWS CLI has multiple modes to choose from depending on your version:

- Legacy retry mode (p. 94)
- Standard retry mode (p. 95)
- Adaptive retry mode (p. 95)

Legacy retry mode

Legacy mode uses an older retry handler that has limited functionality which includes:

- A default value of 4 for maximum retry attempts, making a total of 5 call attempts. This value can be overwritten through the `max_attempts` configuration parameter.
- DynamoDB has a default value of 9 for maximum retry attempts, making a total of 10 call attempts. This value can be overwritten through the `max_attempts` configuration parameter.
- Retry attempts for the following limited number of errors/exceptions:
  - General socket/connection errors:
    - `ConnectionError`
    - `ConnectionClosedError`
    - `ReadTimeoutError`
    - `EndpointConnectionError`
  - Service-side throttling/limit errors and exceptions:
    - `Throttling`
    - `ThrottlingException`