aws re/start

AWS Command Line Interface (AWS CLI)

What you will learn

At the core of the lesson

You will learn how to do the following:

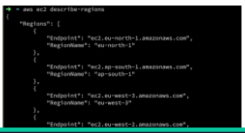
- Define the AWS Command Line Interface (AWS CLI).
- Explain the steps for installing the AWS CLI on Linux.
- Describe details about the AWS CLI.

AWS CLI overview

Three ways to use AWS

AWS Management Console

Provides an intuitive interface for AWS



```
➜  ~ aws ec2 describe-regions
{
    "Regions": [
        {
            "Endpoint": "ec2.eu-north-1.amazonaws.com",
            "RegionName": "eu-north-1"
        },
        {
            "Endpoint": "ec2.ap-south-1.amazonaws.com",
            "RegionName": "ap-south-1"
        },
        {
            "Endpoint": "ec2.eu-west-3.amazonaws.com",
            "RegionName": "eu-west-3"
        },
        {
            "Endpoint": "ec2.eu-west-2.amazonaws.com",
```

AWS CLI

Provides access to AWS services through a Linux, Microsoft Windows, or macOS command line

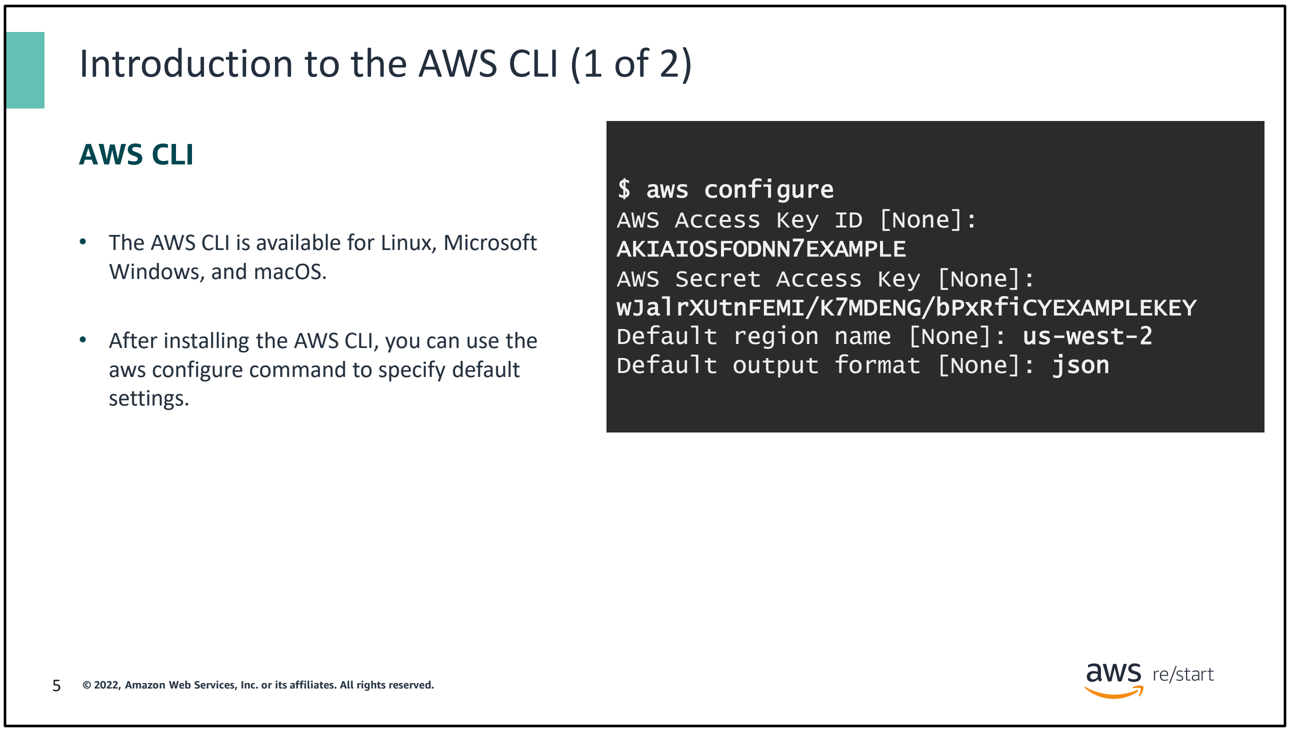
SDKs

Call AWS services APIs from most major programming languages

You can access AWS services in three main ways. All three options are built on a common, REST-like application programming interface (API) that serves as the foundation of Amazon Web Services (AWS):

- The AWS Management Console provides a rich graphical interface for a majority of the products and services that AWS offers. Occasionally, new features might not have all of their capabilities available through the console when the feature initially launches.
- The AWS CLI provides a suite of utilities that can be run from a command program in Linux, macOS, or Microsoft Windows.
- The software development kits (SDKs) are packages that AWS provides. The SDKs provide access to AWS services by using popular programming languages, such as Python, Ruby, .NET, or Java. The SDKs make it straightforward to use AWS in existing applications. You can also use them to create applications to deploy and monitor complex systems entirely through code.

The AWS CLI and the SDKs provide flexibility. AWS users can use them to create their own tools that are specific to their business. Users can also use these tools to customize existing AWS features. For example, a user might create their own scripts or applications for launching Amazon Elastic Compute Cloud (Amazon EC2) instances. These scripts or applications could enforce the use of a specific set of Amazon Machine Images (AMIs). Alternatively, they might add a standard set of tags to apply to resources that are created in the account.



Introduction to the AWS CLI (1 of 2)

AWS CLI

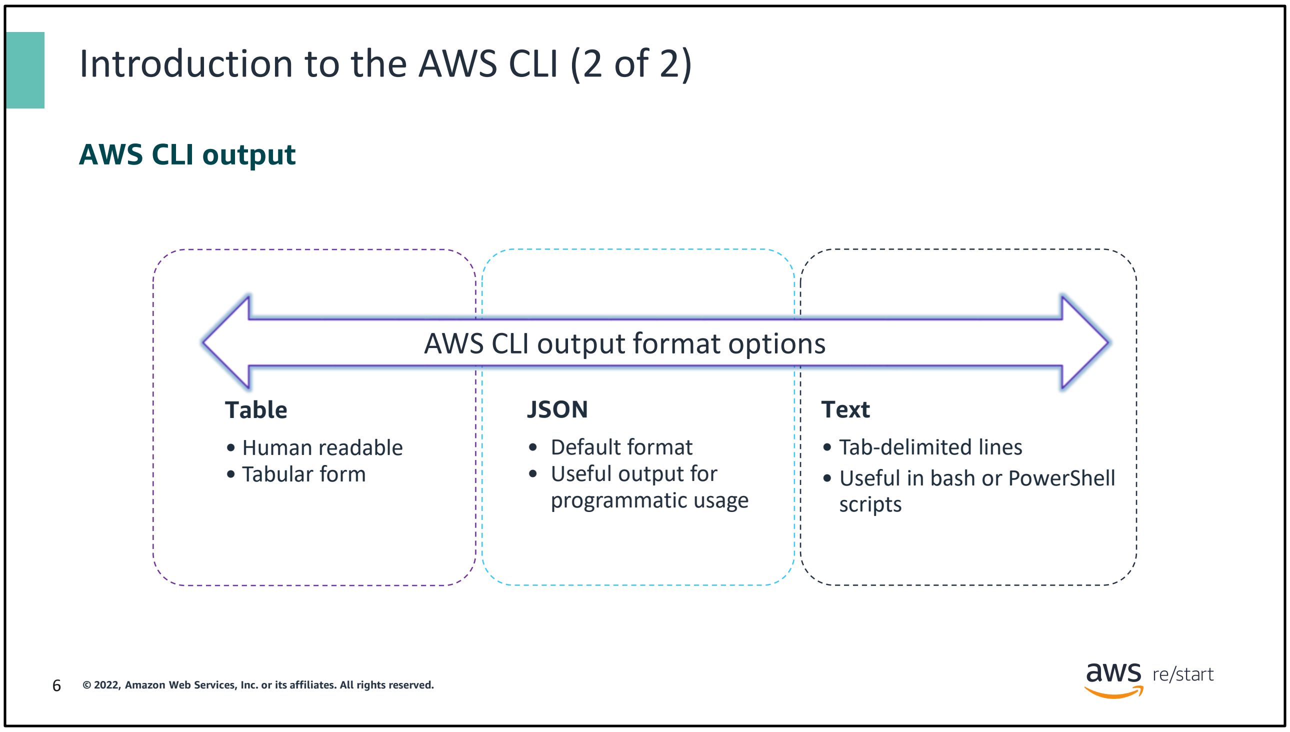
- The AWS CLI is available for Linux, Microsoft Windows, and macOS.
- After installing the AWS CLI, you can use the aws configure command to specify default settings.

```
$ aws configure
AWS Access Key ID [None]:
AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]:
wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: us-west-2
Default output format [None]: json
```

Here is a review of some basic information about the AWS CLI.

The AWS CLI is available for use on Linux, Microsoft Windows, and macOS platforms. It can also be installed on EC2 instances in Linux and Microsoft Windows, and it comes installed by default on Amazon Linux instances.

After installing the AWS CLI, you can use the `aws configure` command to set some default settings for all AWS CLI commands. The main default settings are the access key ID and the secret access key. These values are associated with the AWS Identity and Access Management (IAM) account that you use to access AWS resources. You can also specify a default Region for all commands.



Introduction to the AWS CLI (2 of 2)

AWS CLI output

AWS CLI output format options

Table
- Human readable
- Tabular form

JSON
- Default format
- Useful output for programmatic usage

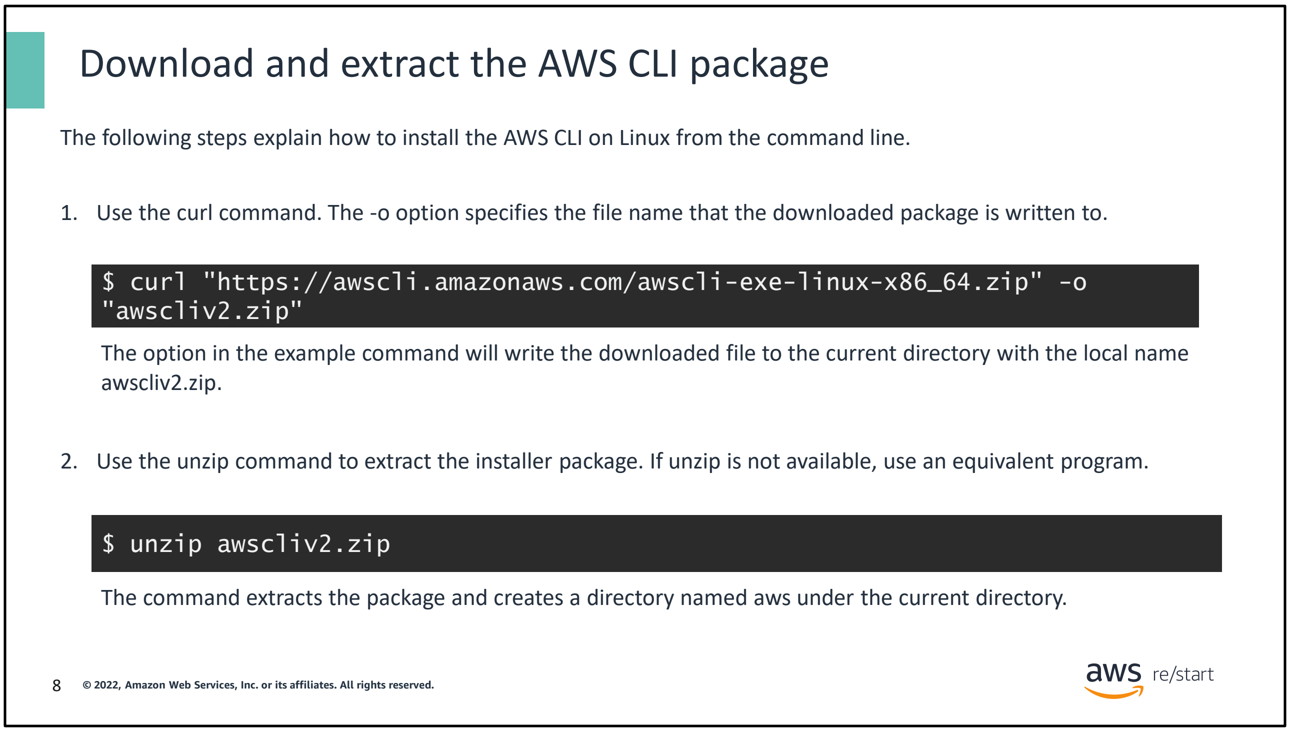
Text
- Tab-delimited lines
- Useful in bash or PowerShell scripts

The default output data format is a JavaScript Object Notation (JSON) document, which is best used in situations where the output will be handled programmatically. JSON is a standard for data formatting and interchange on the internet that can be both read by humans and parsed by machines. Nearly all major programming languages and frameworks provide a way to convert JSON responses into objects or associative arrays.

However, the AWS CLI also supports two other output formats:
- ASCII-formatted table: The table output format is the most readable format because the data has human-readable representations within a tabular form. To specify this output format in a Linux instance, run the following command: `$ export AWS_DEFAULT_OUTPUT="table"`
- Tab-delimited text: This text format displays data as text in tab-delimited lines, and it works well with Unix text processing tools, such as PowerShell scripts. To specify this output format in a Linux instance, run the following command: `$ export AWS_DEFAULT_OUTPUT="text"`

For more information, see “Controlling Command Output from the AWS CLI” at https://docs.aws.amazon.com/cli/latest/userguide/cli-usage-output.html.

Install the AWS CLI



Download and extract the AWS CLI package

The following steps explain how to install the AWS CLI on Linux from the command line.

1. Use the curl command. The -o option specifies the file name that the downloaded package is written to.

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

   The option in the example command will write the downloaded file to the current directory with the local name awscliv2.zip.

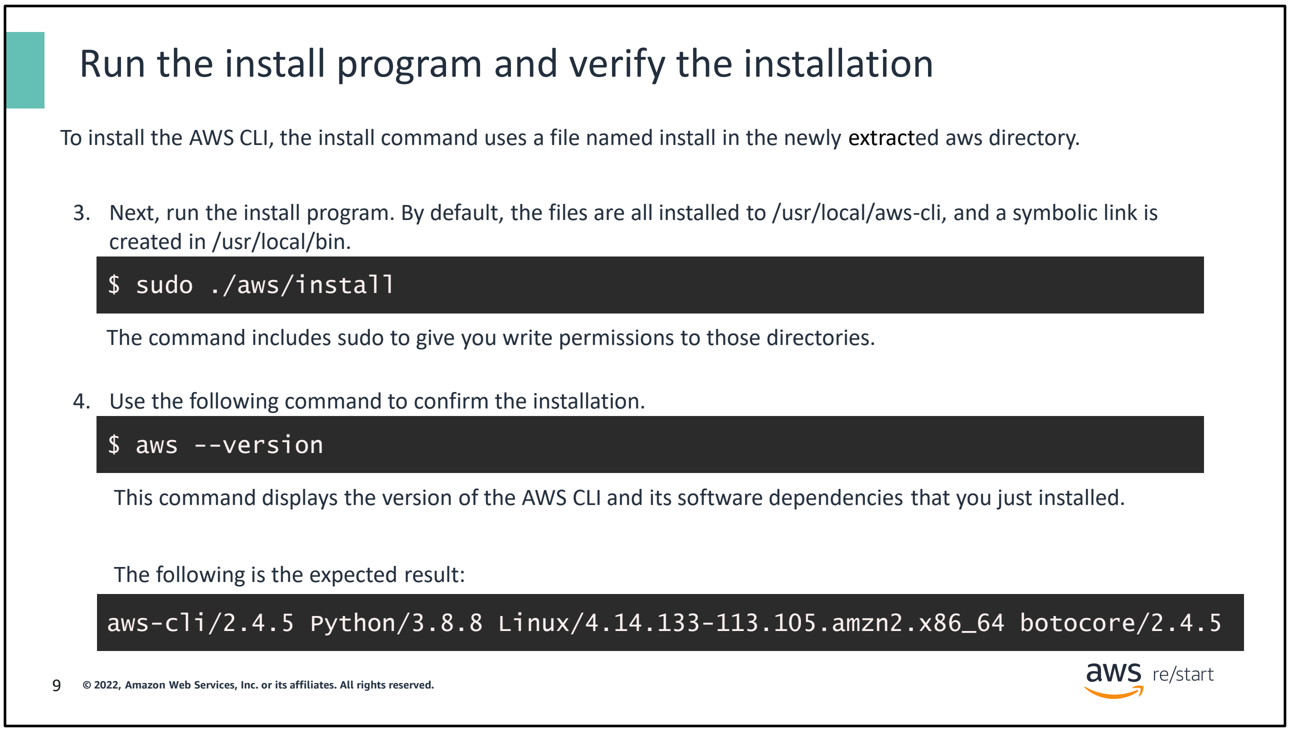
2. Use the unzip command to extract the installer package. If unzip is not available, use an equivalent program.

```
$ unzip awscliv2.zip
```

   The command extracts the package and creates a directory named aws under the current directory.

To install the AWS CLI (Linux), use the following steps:
1. Use the curl command. The `-o` option specifies the file name to which the downloaded package is written. With the given option, the command will write the downloaded file to the current directory with the local name `awscliv2.zip`.

2. Next, use the unzip command to extract the installer package. By default, the files are extracted to a directory named aws under the current directory. You can extract the CLI package by using either the built-in unzip package or an equivalent program.



Run the install program and verify the installation

To install the AWS CLI, the install command uses a file named install in the newly extracted aws directory.

3. Next, run the install program. By default, the files are all installed to /usr/local/aws-cli, and a symbolic link is created in /usr/local/bin.

```
$ sudo ./aws/install
```

The command includes sudo to give you write permissions to those directories.

4. Use the following command to confirm the installation.

```
$ aws --version
```

This command displays the version of the AWS CLI and its software dependencies that you just installed.

The following is the expected result:

```
aws-cli/2.4.5 Python/3.8.8 Linux/4.14.133-113.105.amzn2.x86_64 botocore/2.4.5
```

3. Next, run the following command to install the AWS CLI.

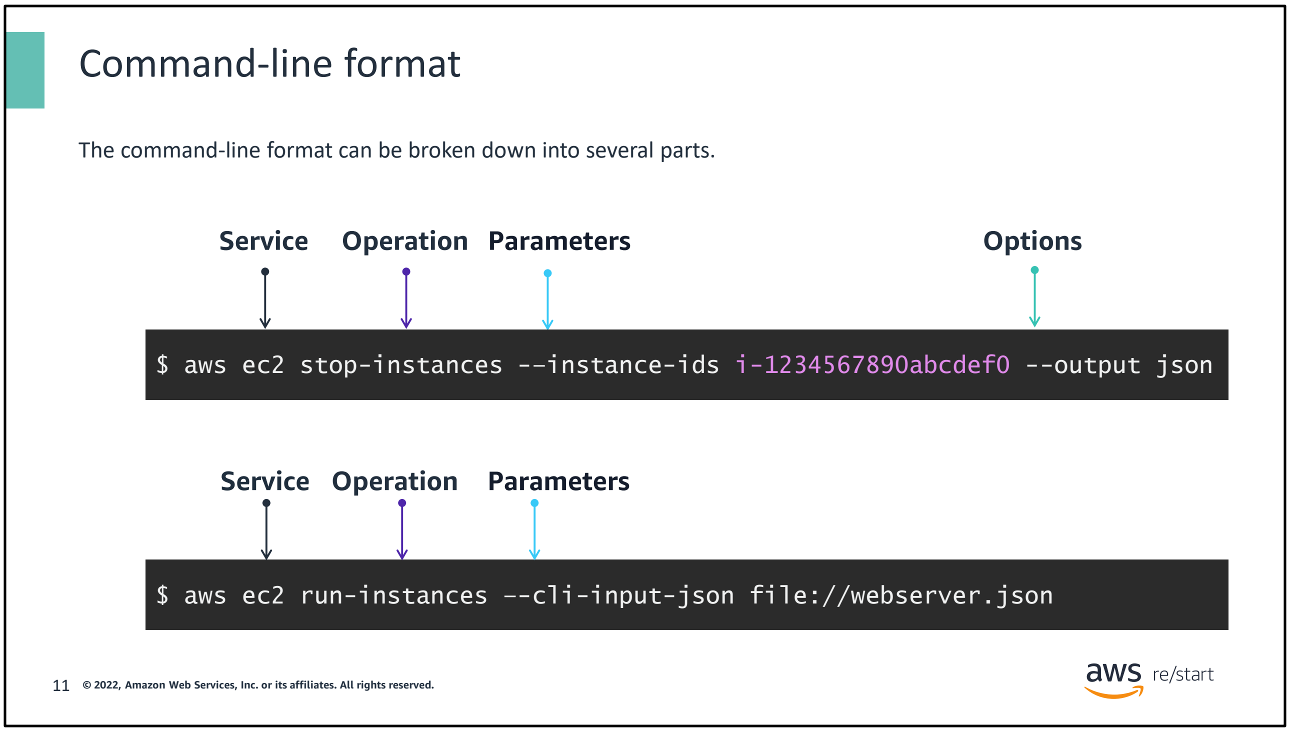
```
$ sudo ./aws/install
```

4. Afterward, run the following command to confirm that the installation has been completed:

```
$ aws --version
```

For more information about troubleshooting and commands, see “Installing or Updating the Latest Version of the AWS CLI” at https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html.

Using the AWS CLI



Command-line format

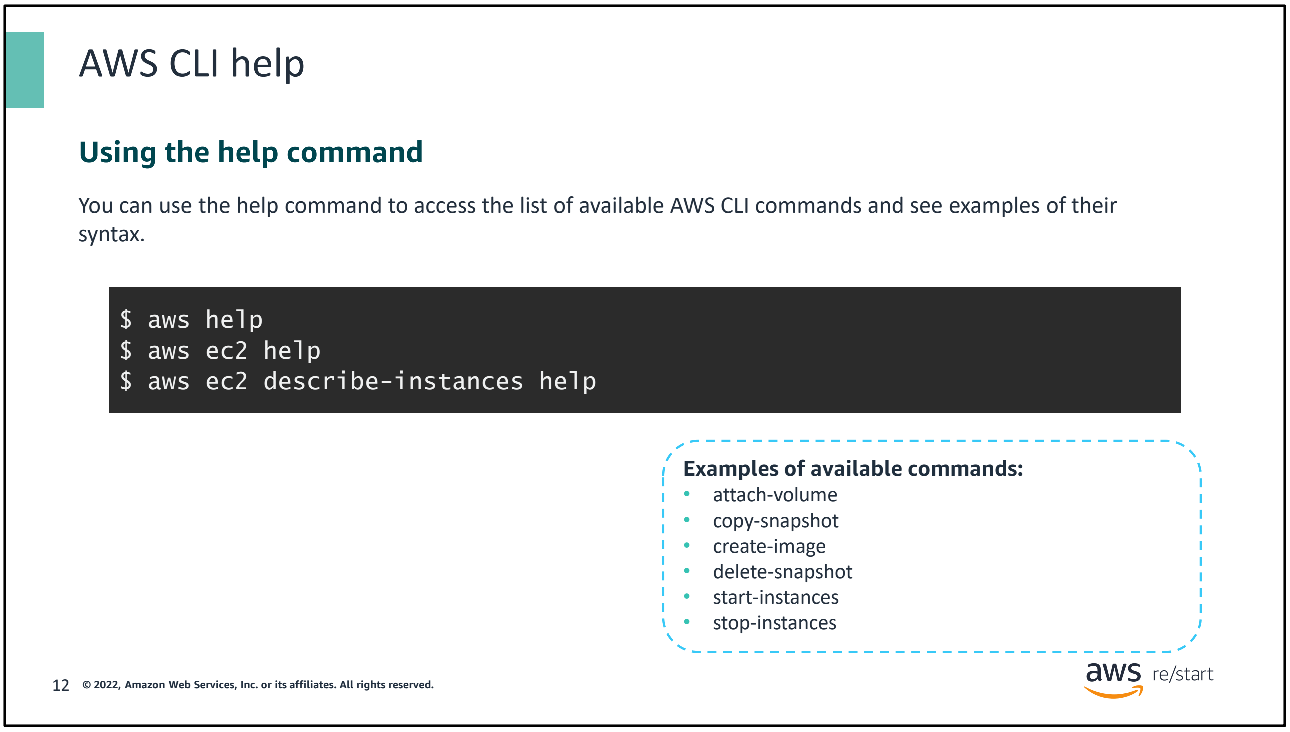
The command-line format can be broken down into several parts.

```
$ aws ec2 stop-instances --instance-ids i-1234567890abcdef0 --output json
```

```
$ aws ec2 run-instances --cli-input-json file://webserver.json
```

The command line format can be broken down into several parts:

1. Use the `aws` command to invoke the AWS CLI program.
2. Specify the top-level service to be called. For example, to invoke Amazon EC2, you enter the `ec2 command` as shown in the example. For more information and a list of supported services, see the `aws` command page on the AWS CLI Command Reference website at https://awscli.amazonaws.com/v2/documentation/api/latest/reference/index.html.
3. Specify the operation to be called. This operation will be performed on the service specified in step 2, such as `stop-instances` or `run-instances`. In the second example on the slide, the `run-instances` subcommand is used to request the creation of a new EC2 instance.
4. Specify the parameters. One or more parameters can be specified, depending on the operation that is called. Parameters are arguments or details that are used to perform the operation. Some operations have required parameters while other operations do not require parameters. Most operations also offer a few or many optional parameters. In the first example, the `stop-instances` operation requires an `instance-ids` parameter. (Otherwise, it would be unclear which instance or instances should be stopped.) Parameter names are preceded by two dashes (`--`). You can also put all of the parameters for an operation in a JSON file and specify the file name as a single parameter. The second example shows this technique.
5. Finally, specify the options. For example, after calling `aws ec2 stop-instances`, use the `--output` option to specify the format of the response.



AWS CLI help

Using the help command

You can use the help command to access the list of available AWS CLI commands and see examples of their syntax.

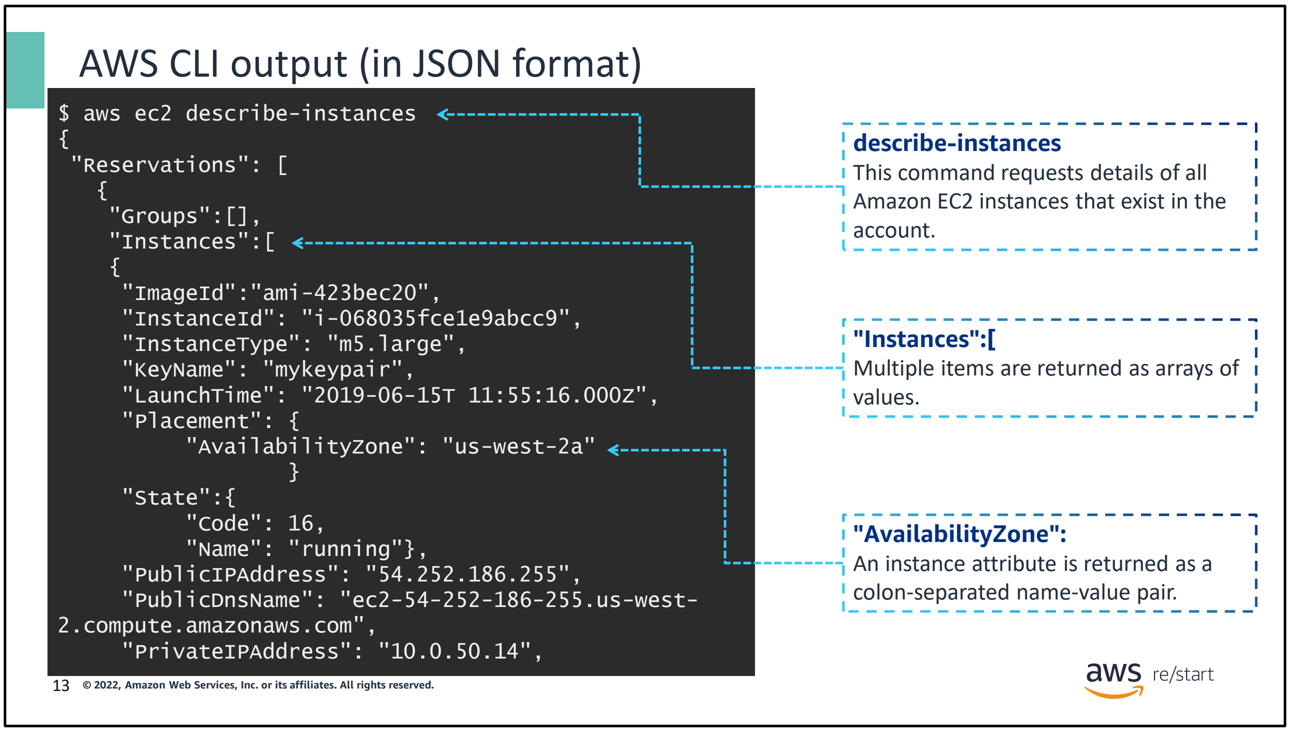
```
$ aws help
$ aws ec2 help
$ aws ec2 describe-instances help
```

Examples of available commands:
- attach-volume
- copy-snapshot
- create-image
- delete-snapshot
- start-instances
- stop-instances

You can use the `help` command to access the list of available AWS CLI commands and see examples of their syntax.

Example: `$ aws ec2 help`
Name: `ec2 –`
Description: Amazon Elastic Compute Cloud (Amazon EC2) provides secure and resizable computing capacity in the AWS Cloud. Using Amazon EC2 eliminates the need to invest in hardware up front, so you can develop and deploy applications faster.
Available commands:
- attach-volume
- copy-snapshot
- create-image
- delete-snapshot
- start-instances
- stop-instances

For more information on the `help` command, see “Getting Help with the AWS CLI” at https://docs.aws.amazon.com/cli/latest/userguide/cli-usage-help.html.



AWS CLI output (in JSON format)

```
$ aws ec2 describe-instances
{
 "Reservations": [
   {
    "Groups":[],
    "Instances":[
    {
     "ImageId":"ami-423bec20",
     "InstanceId": "i-068035fce1e9abcc9",
     "InstanceType": "m5.large",
     "KeyName": "mykeypair",
     "LaunchTime": "2019-06-15T 11:55:16.000Z",
     "Placement": {
          "AvailabilityZone": "us-west-2a"
                  }
     "State":{
          "Code": 16,
          "Name": "running"},
     "PublicIPAddress": "54.252.186.255",
     "PublicDnsName": "ec2-54-252-186-255.us-west-
2.compute.amazonaws.com",
     "PrivateIPAddress": "10.0.50.14",
```

describe-instances
This command requests details of all Amazon EC2 instances that exist in the account.

"Instances":[
Multiple items are returned as arrays of values.

"AvailabilityZone":
An instance attribute is returned as a colon-separated name-value pair.

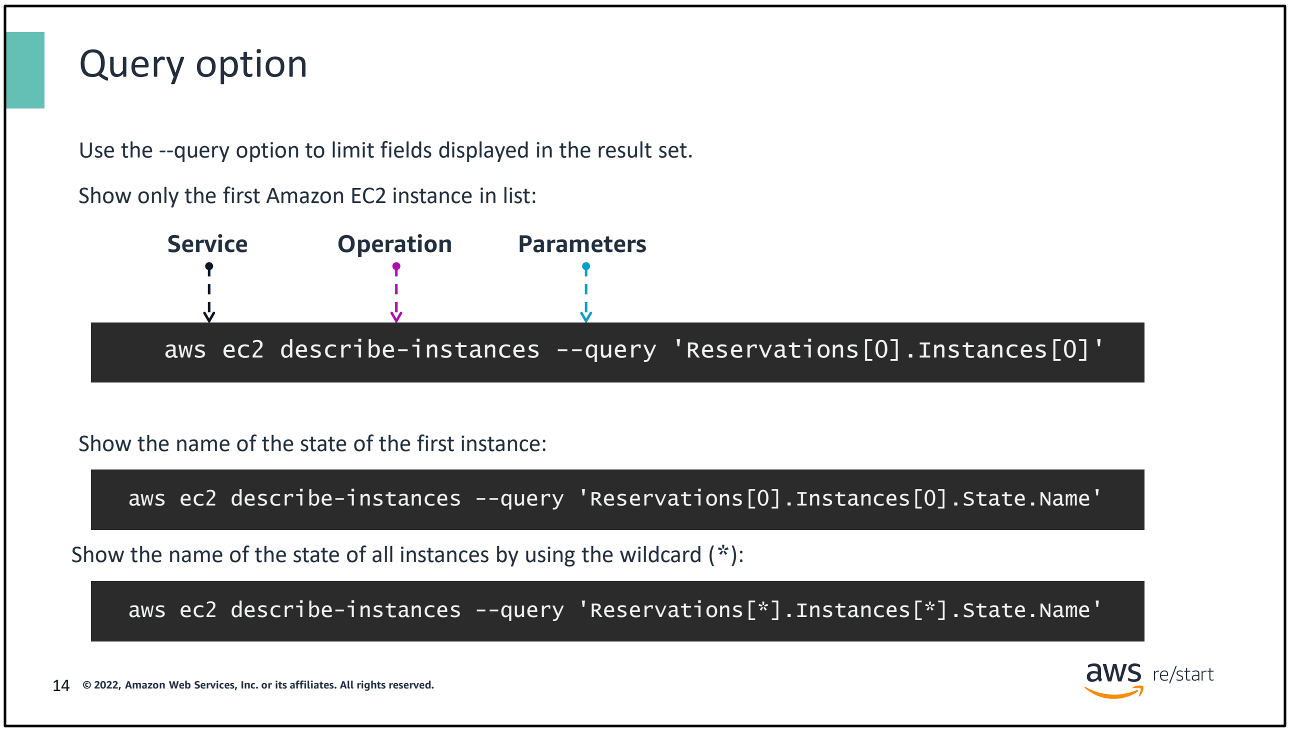
aws re/start

The first line shows an AWS CLI command that was just run (`aws ec2 describe-instances`). The rest of the lines show the result of the AWS CLI command.

The results of the `aws ec2 describe-instances` command are returned as an array of reservations. Each reservation has an array of one or more instances that are associated with it. Both reservations and instances are returned by using JSON array syntax. Later, this course will discuss using array notation to filter responses.

The properties associated with each instance are returned as name-value pairs in quotation marks and are separated by colons. These properties can include the public Domain Name System (DNS) name, the instance state, and others.

For more information about the `aws ec2 describe-instances` command, see the describe-instances command page on the AWS CLI Command Reference website at https://awscli.amazonaws.com/v2/documentation/api/latest/reference/ec2/describe-instances.html.



Query option

Use the --query option to limit fields displayed in the result set.

Show only the first Amazon EC2 instance in list:

Service **Operation** **Parameters**

```
aws ec2 describe-instances --query 'Reservations[0].Instances[0]'
```

Show the name of the state of the first instance:

```
aws ec2 describe-instances --query 'Reservations[0].Instances[0].State.Name'
```

Show the name of the state of all instances by using the wildcard (*):

```
aws ec2 describe-instances --query 'Reservations[*].Instances[*].State.Name'
```

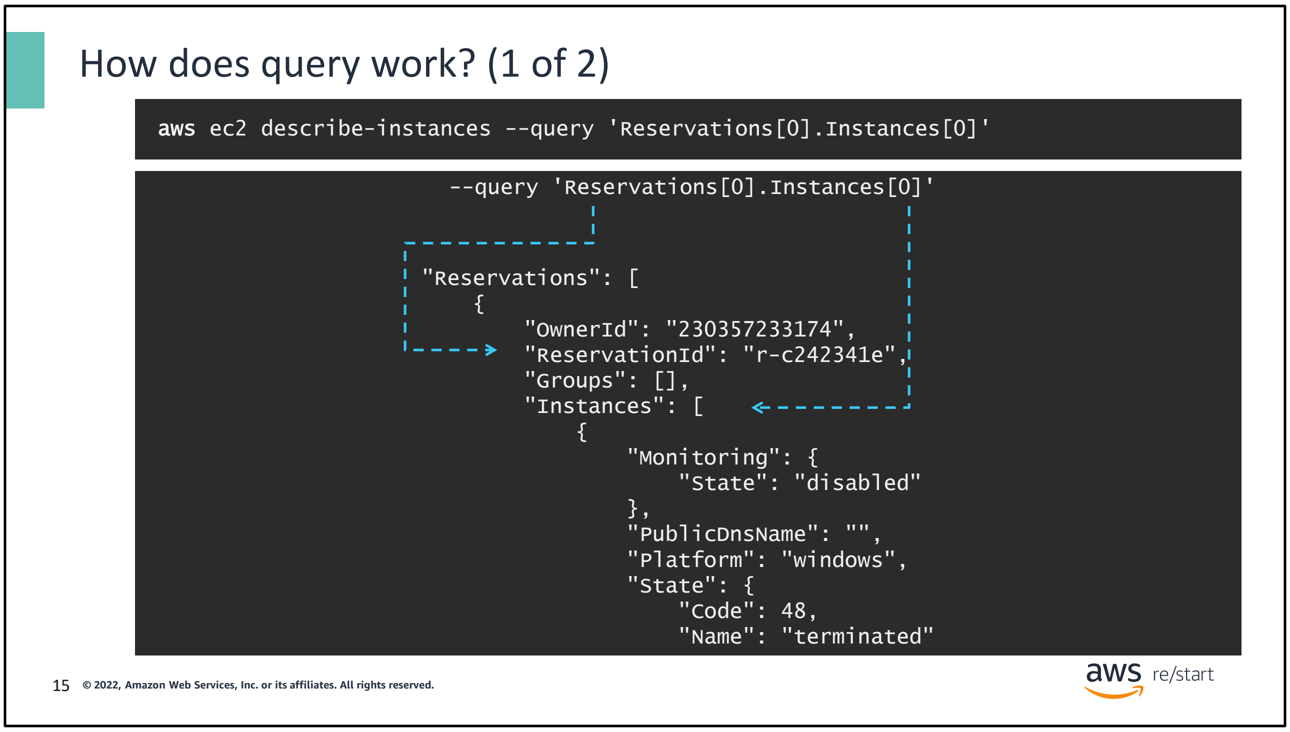
The `--query` option can optionally be included in most commands to limit the results that are returned. Results are filtered on the client side.

You should format queries according to the JMESPath specification, which defines a syntax for searching JSON documents. For more information, see “JMESPath Specification” on the JMESPath website at https://jmespath.org/specification.html.

Queries are constructed by designating the subsection of the response that you want to see. You can specify the subsection by providing the full path to that section, starting from the top-most element of the response.

Array elements can be referenced using zero-based indexing. For example, to describe only the first instance returned by the `describe-instances` operation, use the path `Reservations[0].Instances[0]` in the query. This path says, “For the first reservation, return the details about the first instance.” You could further specify that you want to see only specific attributes of this instance, such as its current operating state.

For another example, to view the states of all instances, you can use wildcards instead of indexes to return results from all reservations and all instances. The `[*]` syntax can be used in any array to specify that you want to view the specified property or child element for all members of that array.



How does query work? (1 of 2)

```
aws ec2 describe-instances --query 'Reservations[0].Instances[0]'
```

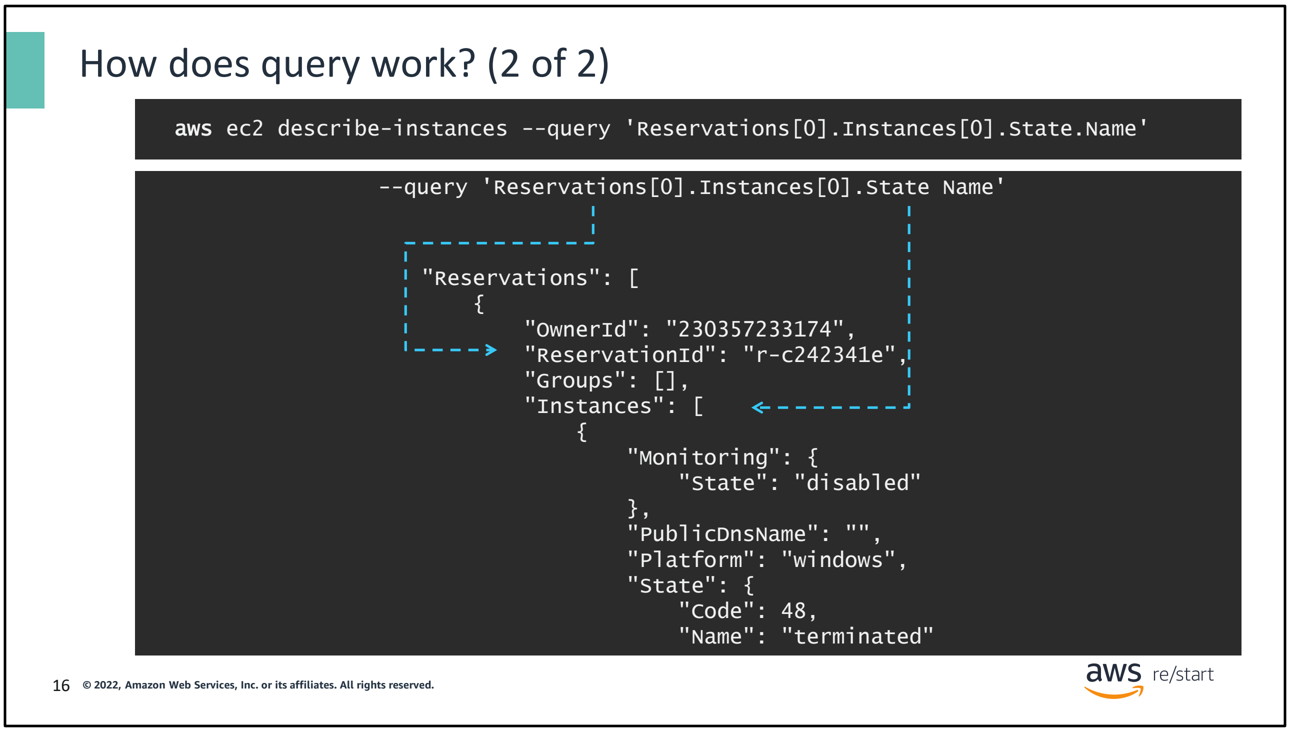
```
--query 'Reservations[0].Instances[0]'

"Reservations": [
    {
        "OwnerId": "230357233174",
        "ReservationId": "r-c242341e",
        "Groups": [],
        "Instances": [
            {
                "Monitoring": {
                    "State": "disabled"
                },
                "PublicDnsName": "",
                "Platform": "windows",
                "State": {
                    "Code": 48,
                    "Name": "terminated"
```

The `--query` option limits what is displayed from the returned results.

In the example on the slide, the `aws ec2 describe-instances command` is called with the following `--query` option:

- The query specifies `Reservations[0]`. Thus, only the first reservation is returned.
- The query specifies `Instances[0]`. Thus, only the details about the first EC2 instance in the first reservation are returned.



How does query work? (2 of 2)

```
aws ec2 describe-instances --query 'Reservations[0].Instances[0].State.Name'
```

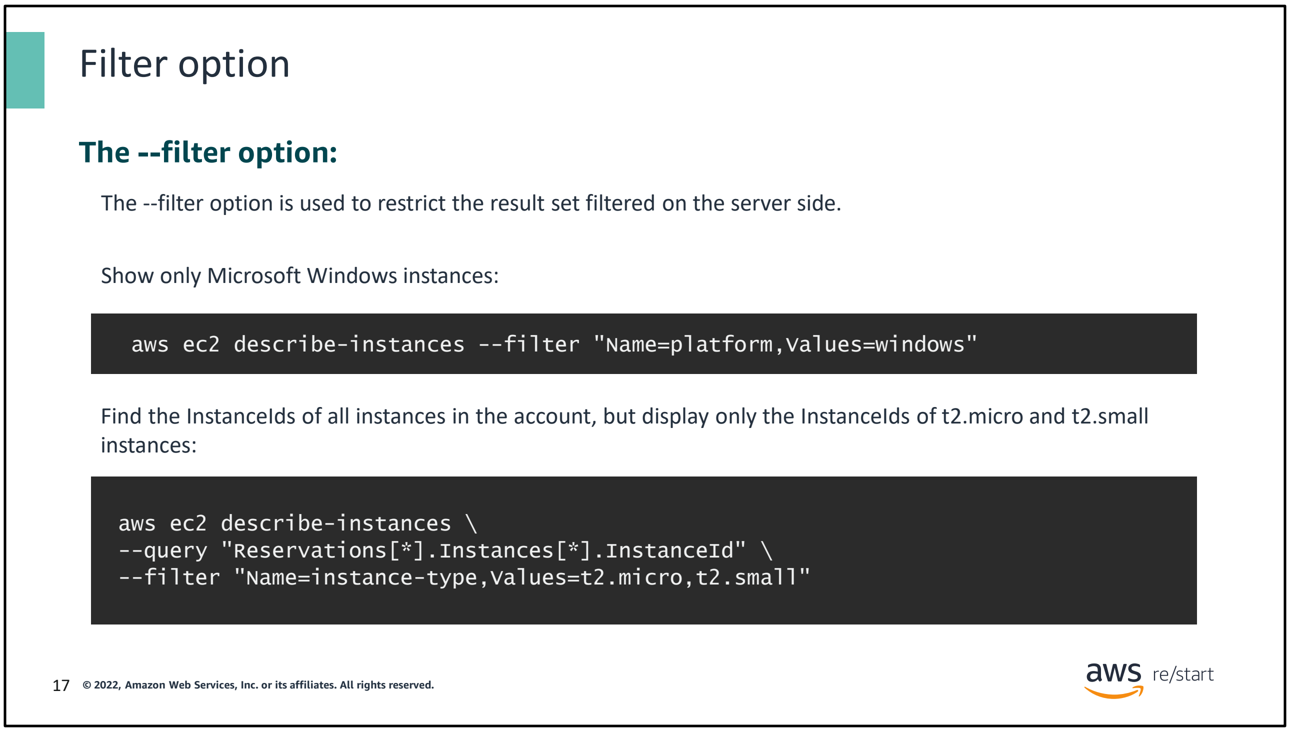
```
--query 'Reservations[0].Instances[0].State Name'

 "Reservations": [
     {
         "OwnerId": "230357233174",
         "ReservationId": "r-c242341e",
         "Groups": [],
         "Instances": [
             {
                 "Monitoring": {
                     "State": "disabled"
                 },
                 "PublicDnsName": "",
                 "Platform": "windows",
                 "State": {
                     "Code": 48,
                     "Name": "terminated"
```

In this example, the query is more specific than on the previous slide. Here, the query also specifies that the only item to be returned should be the value of the state name of the first instance from the first reservation.

The parts of the JSON document are what matched the query. The actual result set that was returned by running this command would be a single line: `terminated`

When you query a large collection of instances in an account, use the `--query` option. This option is useful for identifying only the instances that match a set of characteristics of interest.



Filter option

The --filter option:

The --filter option is used to restrict the result set filtered on the server side.

Show only Microsoft Windows instances:

```
aws ec2 describe-instances --filter "Name=platform,Values=windows"
```

Find the InstanceIds of all instances in the account, but display only the InstanceIds of t2.micro and t2.small instances:

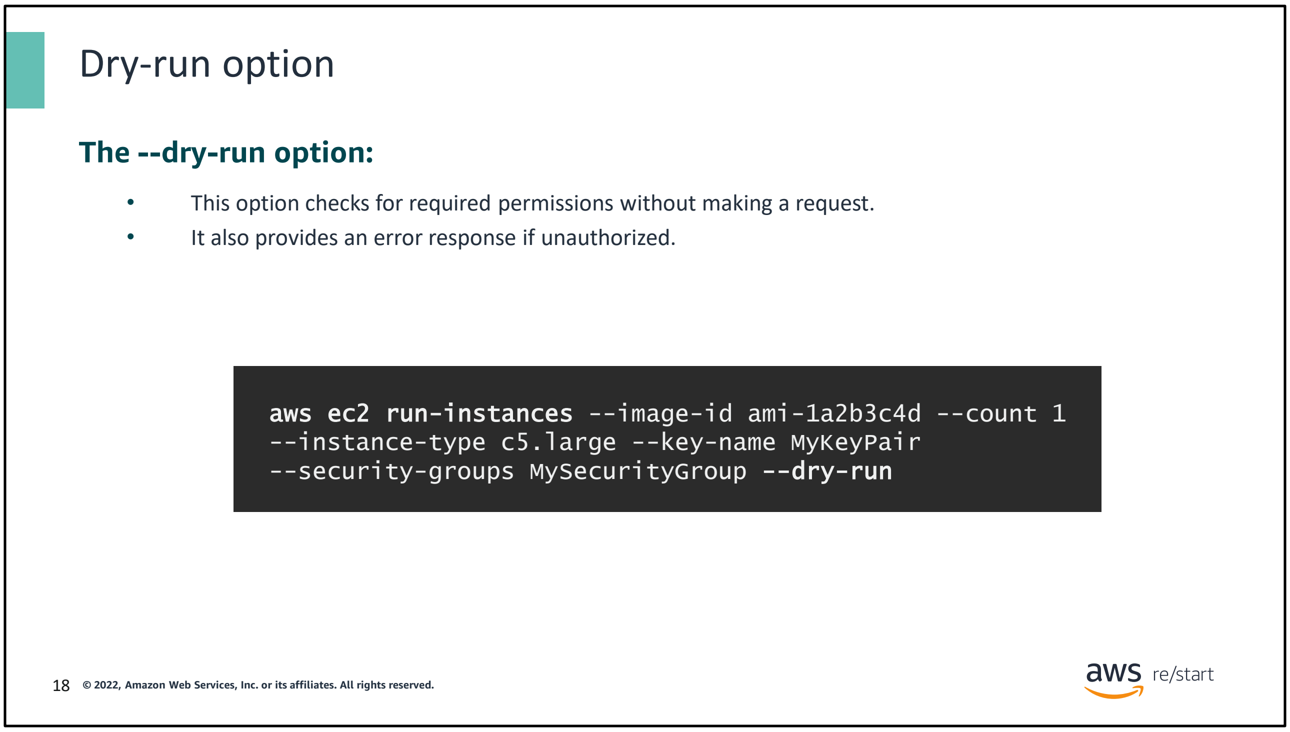
```
aws ec2 describe-instances \
--query "Reservations[*].Instances[*].InstanceId" \
--filter "Name=instance-type,Values=t2.micro,t2.small"
```

The `--filter` option is used to restrict the result set, which is filtered on the server side.

The examples illustrate the use of `the –filter option` with the `aws ec2 describe-instances` command.

In the first example, the returned results are filtered so that only instances that run Windows Server are displayed.

The second example shows a command that includes both the `–query` option and the `–filter` option. This command finds the `InstanceIds` of all instances in the account but then displays only the `InstanceIds` of the `t2.micro` and `t2.small` instances.



Dry-run option

The --dry-run option:

- This option checks for required permissions without making a request.
- It also provides an error response if unauthorized.

```
aws ec2 run-instances --image-id ami-1a2b3c4d --count 1
--instance-type c5.large --key-name MyKeyPair
--security-groups MySecurityGroup --dry-run
```

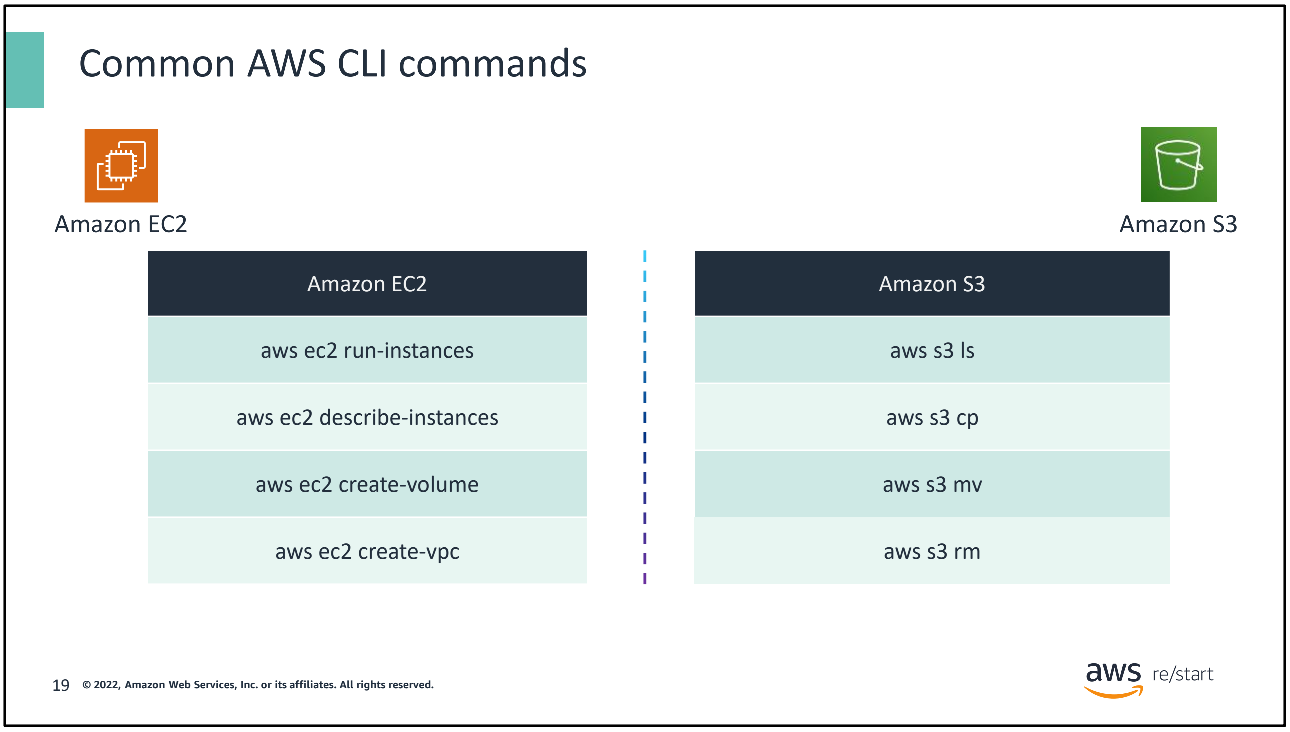
Another CLI option is `the --dry-run option`. It is used for testing purposes because it checks whether the required permissions for the action are present without actually making the request.

If the operation is authorized, the command returns the following message:

An error occurred (DryRunOperation) when calling the <operationName> operation: Request would have succeeded, but DryRun flag is set.

If the operation is not authorized, the command returns the following message:

An error occurred (UnauthorizedOperation) when calling the <operationName> operation: You are not authorized to perform this operation.



Common AWS CLI commands

Amazon EC2

Amazon S3

Amazon EC2	Amazon S3
aws ec2 run-instances	aws s3 ls
aws ec2 describe-instances	aws s3 cp
aws ec2 create-volume	aws s3 mv
aws ec2 create-vpc	aws s3 rm

Here is a short list of some common AWS CLI commands. These commands are related to two of the most popular AWS services: Amazon EC2 and Amazon Simple Storage Service (Amazon S3).

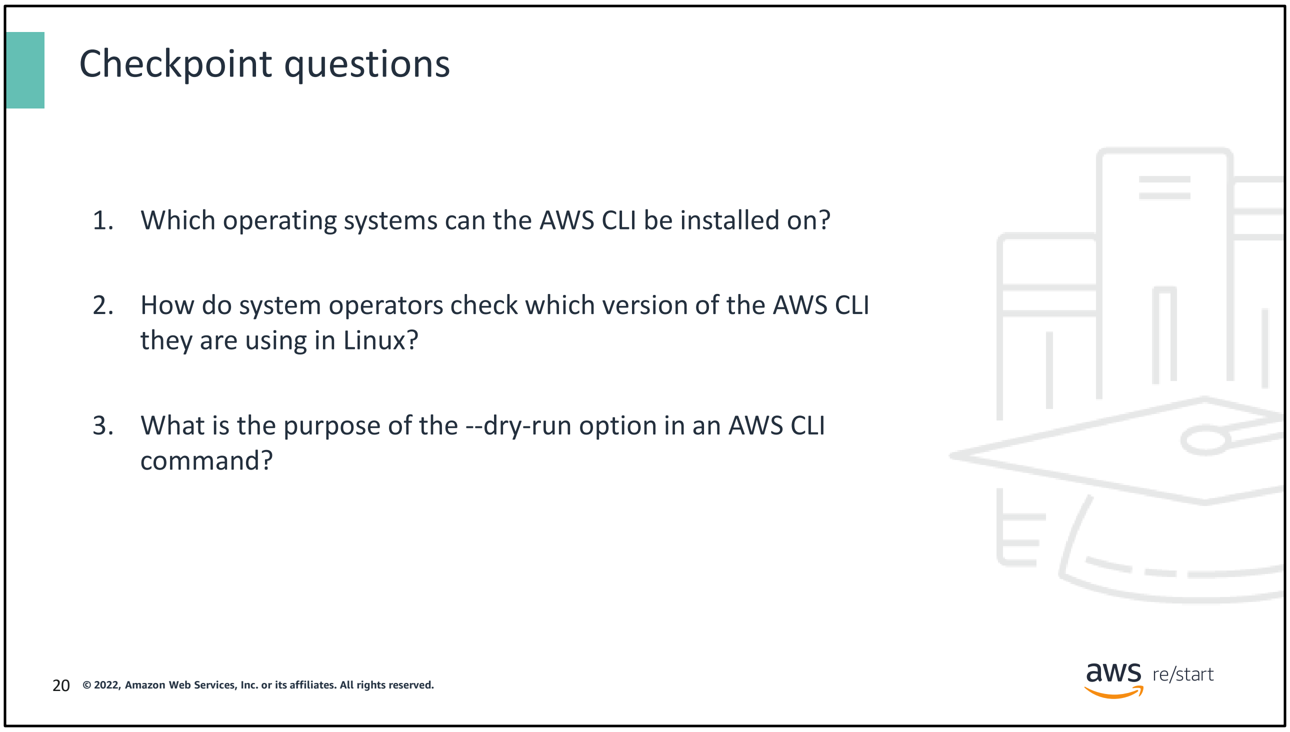
You use the aws ec2 command to accomplish Amazon Virtual Private Cloud (Amazon VPC) and Amazon Elastic Block Store (Amazon EBS) actions.

Here is a brief description of each of the commands that are shown, and their respective actions:

- Commands based on Amazon EC2:
    - `aws ec2 run-instances`: This command launches the specified number of instances from an AMI.
    - `aws ec2 describe-instances`: This command describes any instances that exist in the account.
    - `aws ec2 create-volume`: This command creates an EBS volume that can be attached to an instance in the same Availability Zone.
    - `aws ec2 create-vpc`: This command creates a VPC with the Classless Inter-Domain Routing (CIDR) block specified.
- Commands based on Amazon S3:
    - `aws s3 ls`: This command lists Amazon S3 objects and common prefixes under a prefix or all S3 buckets. Optionally, if you specify a specific bucket in the account, and the `ls` command will list the contents of the

specified bucket.

- `aws s3 cp`: This command copies a file to, from, or between Amazon S3 locations. Use it to copy local files to Amazon S3, files from Amazon S3 to a laptop, or Amazon S3 files to other Amazon S3 locations.
- `aws s3 mv`: This command moves a local file or an Amazon S3 object to another location locally or in Amazon S3.
- `aws s3 rm`: This command deletes an Amazon S3 object.



Checkpoint questions

1. Which operating systems can the AWS CLI be installed on?
2. How do system operators check which version of the AWS CLI they are using in Linux?
3. What is the purpose of the --dry-run option in an AWS CLI command?

aws re/start

1. Which operating systems can the AWS CLI be installed on?

   Operating systems that can install the AWS CLI are as follows:
   - Linux
   - MacOS
   - Windows

2. How do system operators check which version of the AWS CLI they are using in Linux?

   Use the following command to check which version of the AWS CLI has been installed:

   ```
   $ aws –version
   ```

3. What is the purpose of the `--dry-run` option in an AWS CLI command?

   It is used for testing because it checks whether the required permissions for the action are present without actually making the request.

Key ideas

- You can use the AWS CLI to interact with AWS services by using commands entered in a command-line tool.
- The following is the structure of an AWS CLI command:

```
aws <serviceName> <operation>
[options and parameters]
```

- The --filter option runs on the server side and returns results that match a filter condition.
- The --query option runs on the client side and limits how much of what was returned by the server is displayed.
- The --dry-run option checks whether you have the permissions to run a command without actually running the command.

aws re/start

Thank you