

# The AWS Concept Journey

Version 1.0  
April-2024

By Dr. Shlomi Boutnaru



# Table of Contents

<b>Table of Contents</b> .....	<b>2</b>
<b>AWS Region</b> .....	<b>4</b>
<b>AWS Account ID</b> .....	<b>5</b>
<b>Availability Zones (AZ)</b> .....	<b>6</b>
<b>ARN (Amazon Resource Name)</b> .....	<b>7</b>
<b>AWS Management Console</b> .....	<b>8</b>
<b>AWS CLI (Command Line Interface)</b> .....	<b>9</b>
<b>AWS SDK (Software Development Kit)</b> .....	<b>10</b>
<b>AWS CloudShell</b> .....	<b>11</b>

# AWS Region

Overall, an AWS Region is a separate physical location (geographic area) around the world where AWS cluster data centers. Each Region has at least 3 Availability Zones (more on that in a future writeup), which are isolated locations within the Region. This isolation helps to protect applications from disruptions due to the fact each AZ has independent cooling, power and physical security. Also, the AZs in a region are connected ultra-low-latency resilient fiber networks<sup>1</sup>.

Moreover, as of April 2023 there are 27 regions in AWS in 21 distinct countries. Those countries are: Australia, Bahrain, Brazil, Canada, China, France, Germany, India, Indonesia, Ireland, Italy, Japan, Singapore, South Africa, South Korea, Spain, Sweden, Switzerland, United Arab Emirates, United Kingdom and United States<sup>2</sup> - as shown in the map below that is relevant for December 2022<sup>3</sup>.

Thus, we can choose the region that is closest to your users or the one which meets our compliance requirements. We can sum up the benefits of regions into the following points: global reach, isolation, compliance and high availability<sup>4</sup>.

Lastly, in order to use a specific AWS region we just need to log to our AWS account and select the one we want. After that all of our resources are going to be launched in that specific region and not in other ones<sup>5</sup>.



---

<sup>1</sup> [https://aws.amazon.com/about-aws/global-infrastructure/regions\\_az/](https://aws.amazon.com/about-aws/global-infrastructure/regions_az/)

<sup>2</sup> <https://cloudregions.io/aws/regions>

<sup>3</sup> <https://docs.aws.amazon.com/whitepapers/latest/aws-fault-isolation-boundaries/regions.html>

<sup>4</sup> <https://docs.aws.amazon.com/whitepapers/latest/aws-fault-isolation-boundaries/regions.html>

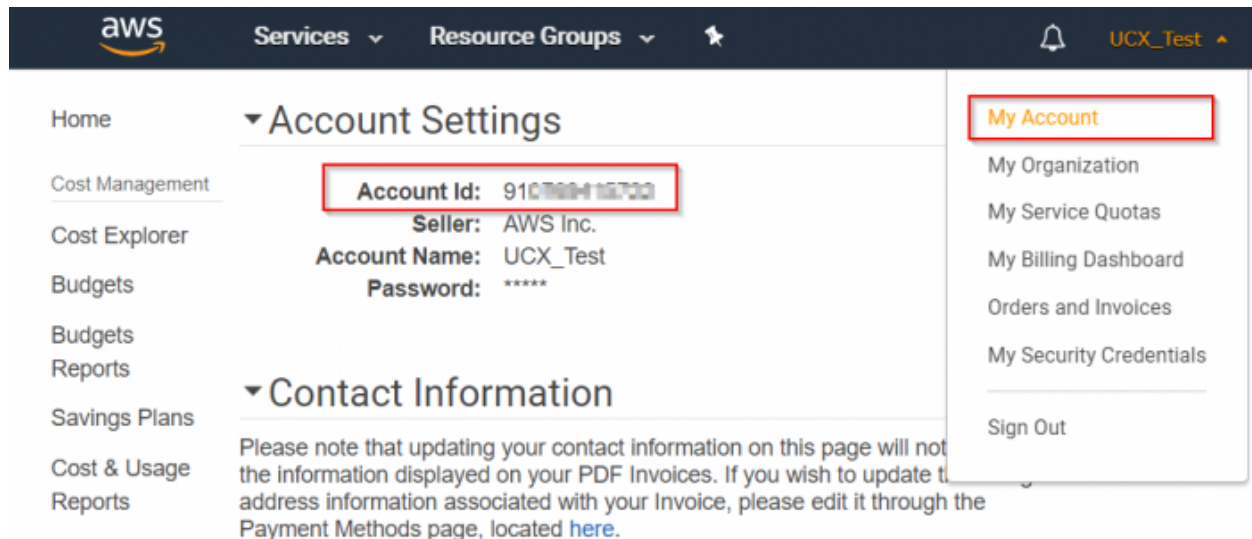
<sup>5</sup> <https://cloudacademy.com/blog/aws-global-infrastructure/>

# AWS Account ID

In general, an AWS account ID is a unique identifier which is used in order to track the usage/bill of AWS services and provide a boundary for granting access to cloud resources. It is also used for a reference when contacting the AWS support<sup>6</sup>.

Moreover, it can be used as a subdomain for our own sign-in page URL in the following format: [https://\[AWS\\_Account\\_ID\].signin.aws.amazon.com/console/](https://[AWS_Account_ID].signin.aws.amazon.com/console/), this is known as account alias. We can create it using the AWS CLI (“aws iam create-account-alias”) or from the management console<sup>7</sup>.

Lastly, we can retrieve our AWS account ID using AWS CLI by calling “get-caller-identity”<sup>8</sup> or by using the AWS management console in the “My Account” page - as shown in the screenshot below<sup>9</sup>.



<sup>6</sup> <https://docs.aws.amazon.com/marketplace/latest/buyerguide/GettingSupport.html>

<sup>7</sup> [https://docs.aws.amazon.com/IAM/latest/UserGuide/console\\_account-alias.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/console_account-alias.html)

<sup>8</sup> <https://docs.aws.amazon.com/cli/latest/reference/sts/get-caller-identity.html>

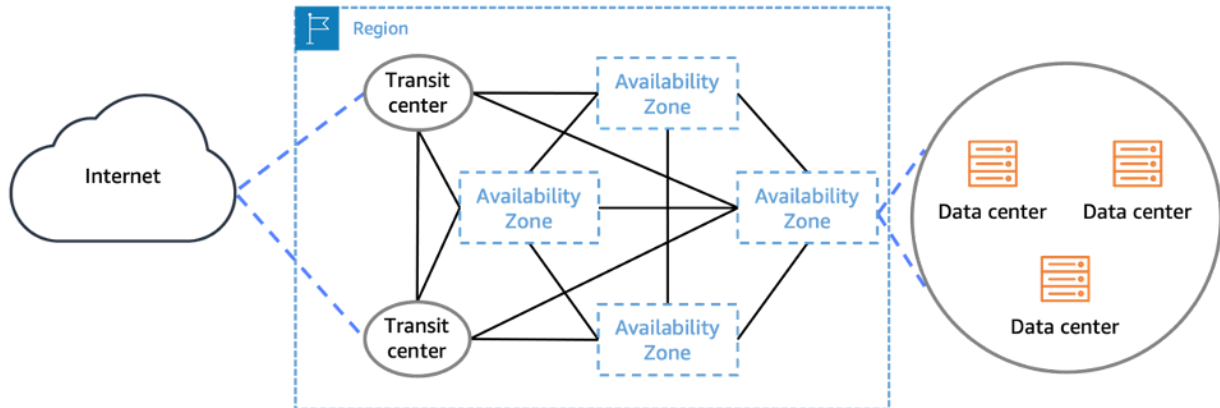
<sup>9</sup> <https://www.emetrotel.com/tsd/content/installing-ucx-your-aws-account>

# Availability Zones (AZ)

In general, an Availability Zone (AZ) is a distinct location within an AWS Region<sup>10</sup> that is built in the sense of isolation from failures in other AZs. Based on the AWS documentation there are today 96 AZs within 30 regions (this number probably is going to change). We can describe an AZ as one or more distinct data centers in an AWS region. Each AZ has its own networking, power infrastructure, cooling and connectivity. Also, AZs are meaningfully distant from each other, up to 60 miles (~100 km) to prevent correlated failures, but close enough to use synchronous replication with single-digit millisecond latency<sup>11</sup>.

Thus, the isolation helps to protect applications from disruptions caused by hardware failures, power outages, and other events. We can summarize the benefits of AZs to the following points: high availability, fault tolerance, scalability and performance<sup>12</sup>. When picking an AZ (it is true also for a region) we should consider: cost, compliance, service availability and distance<sup>13</sup>.

Moreover, all AZs in an AWS Region are interconnected with high-bandwidth, low-latency networking, over fully redundant, dedicated metro fiber between AZs<sup>14</sup>. Lastly, Availability Zones consist of one or more physical data centers that are redundantly connected to each other and the internet - as shown in the diagram below<sup>15</sup>.



<sup>10</sup> <https://medium.com/@boutnaru/the-aws-journey-aws-region-52c28ca4ac84>

<sup>11</sup> <https://docs.aws.amazon.com/whitepapers/latest/aws-fault-isolation-boundaries/availability-zones.html>

<sup>12</sup> <https://www.lightlytics.com/resources/availability-zone>

<sup>13</sup> <https://www.opcito.com/blogs/aws-regions-availability-zones-and-strategy-best-practices>

<sup>14</sup> [https://aws.amazon.com/about-aws/global-infrastructure/regions\\_az/](https://aws.amazon.com/about-aws/global-infrastructure/regions_az/)

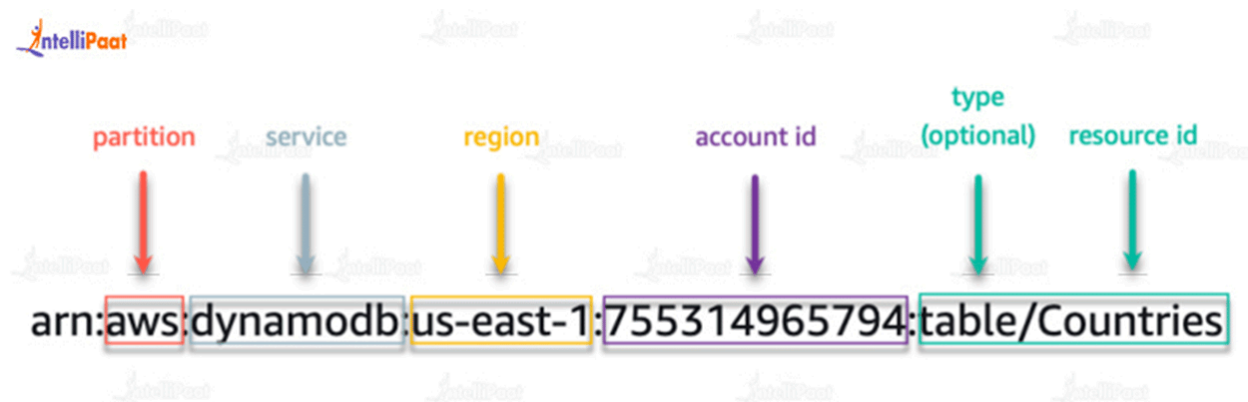
<sup>15</sup> <https://docs.aws.amazon.com/whitepapers/latest/aws-fault-isolation-boundaries/availability-zones.html>

# ARN (Amazon Resource Name)

The goal of an ARN (Amazon Resource Name) is to uniquely identify an AWS resource. ARN has a specific format which is dependent on the resource, it is important to know that they are resources which omit the account ID/partition/region/both as part of their ARN<sup>16</sup> - as shown in the example below<sup>17</sup>.

Overall, there are three different ARN formats: “arn:aws:service:region:account-id:resource-id”, “arn:aws:service:region:account-id:resource-type/resource-id” or “arn:aws:service:region:account-id:resource-type:resource-id”<sup>18</sup>. By the way, the partition which reflects the AWS region to which the ARN is linked is most of the time “aws” (as shown above) but can also be “aws-cn” or “aws-us-gov”.

Thus, examples of ARNs are: “arn:aws:iam::133713371337:user/troller” (IAM user) and “arn:aws:ec2:us-east-1:133713371337:vpc/vpc-0f9801d177TROLLER” (VPC), by the way the account ID and resource names are just examples and not real information<sup>19</sup>. Also, in certain situations we can use wildcards (\* and ?) within an ARN<sup>20</sup>. Lastly, we can summarize that an ARN helps us in uniquely identifying an AWS resource<sup>21</sup>.



<sup>16</sup> <https://docs.aws.amazon.com/IAM/latest/UserGuide/reference-arns.html>

<sup>17</sup> <https://intellipaat.com/blog/arn-amazon-resource-name/>

<sup>18</sup> <https://docs.aws.amazon.com/managedservices/latest/userguide/find-arn.html>

<sup>19</sup> <https://docs.aws.amazon.com/IAM/latest/UserGuide/reference-arns.html>

<sup>20</sup> <https://docs.aws.amazon.com/quicksight/latest/APIReference/qs-arn-format.html>

<sup>21</sup> <https://sst.dev/archives/what-is-an-arn.html>

# AWS Management Console

The AWS management console is a web interface which can be used for accessing and managing cloud resources/services. As of the start of 2024 there are over 150 services that can be accessed from the management console. Also, the console enables managing and monitoring monthly bills, health, service usage and users<sup>22</sup>.

Overall, we can use the following link in order to logon to the console: <https://console.aws.amazon.com/>. There are two options to logon with a “Root user” or an “IAM user”. The “Root user” is the account owner which can perform the most privileged tasks, we need to provide the email address we used to enroll the user for logging - as shown in the screenshot below. By the way, when using an “IAM user” we need to provide our account ID.

Lastly, the management console provides secure login and sessions (limiting session lifetime, using the federation API and more), browser support like Chrome, Safari, Firefox, and Edge<sup>23</sup> and there is even a mobile application for iOS/Android devices<sup>24</sup>.



## Sign in

**Root user**  
Account owner that performs tasks requiring unrestricted access. [Learn more](#)

**IAM user**  
User within an account that performs daily tasks. [Learn more](#)

**Root user email address**

By continuing, you agree to the [AWS Customer Agreement](#) or other agreement for AWS services, and the [Privacy Notice](#). This site uses essential cookies. See our [Cookie Notice](#) for more information.



## Sign in

**Root user**  
Account owner that performs tasks requiring unrestricted access. [Learn more](#)

**IAM user**  
User within an account that performs daily tasks. [Learn more](#)

**Account ID (12 digits) or account alias**

By continuing, you agree to the [AWS Customer Agreement](#) or other agreement for AWS services, and the [Privacy Notice](#). This site uses essential cookies. See our [Cookie Notice](#) for more information.

<sup>22</sup> <https://aws.amazon.com/console/>

<sup>23</sup> <https://aws.amazon.com/console/features/>

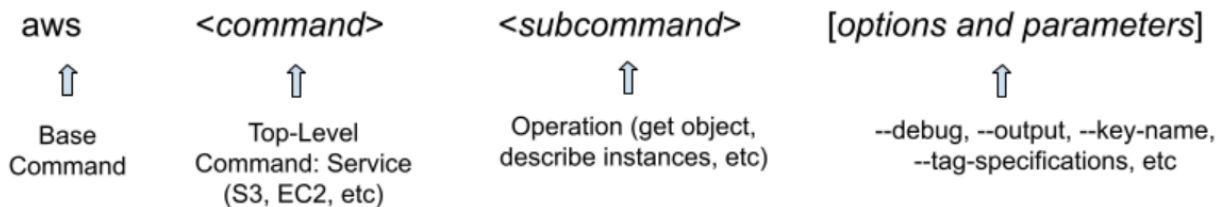
<sup>24</sup> <https://aws.amazon.com/console/mobile/>

# AWS CLI (Command Line Interface)

The AWS CLI (Command Line Interface) allows us to manage our AWS services with automation using scripts or manually from the command line<sup>25</sup>. AWS CLI is has support for Linux shells (like bash/zsh/tcsh/etc), Windows command line (cmd.exe/Powershell.exe), macOS or remotely using EC2 instances via SSH<sup>26</sup>.

Moreover, Using AWS CLI we can perform different functions which are equivalent to that provided by the browser-based AWS Management Console<sup>27</sup>. AWS CLI is open source and described in its repo as “Universal Command Line Interface for Amazon Web Services”, it is written in Python<sup>28</sup>.

Lastly, there are two versions for AWS CLI, it is recommended to use version 2.0. However, you should know that they are breaking changes between AWS CLI version 1 and version 2<sup>29</sup>. I also suggest going over a cheat sheet for basic AWS CLI commands. By the way, the format of a regular command is shown below<sup>30</sup>.



---

<sup>25</sup> <https://aws.amazon.com/cli/>

<sup>26</sup> <https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-welcome.html>

<sup>27</sup> <https://medium.com/@boutnaru/the-aws-journey-aws-management-console-6de0524bb548>

<sup>28</sup> <https://github.com/aws/aws-cli>

<sup>29</sup> <https://docs.aws.amazon.com/cli/latest/userguide/cliv2-migration-changes.html>

<sup>30</sup> <https://plainenglish.io/blog/aws-cli-commands-cheatsheet>

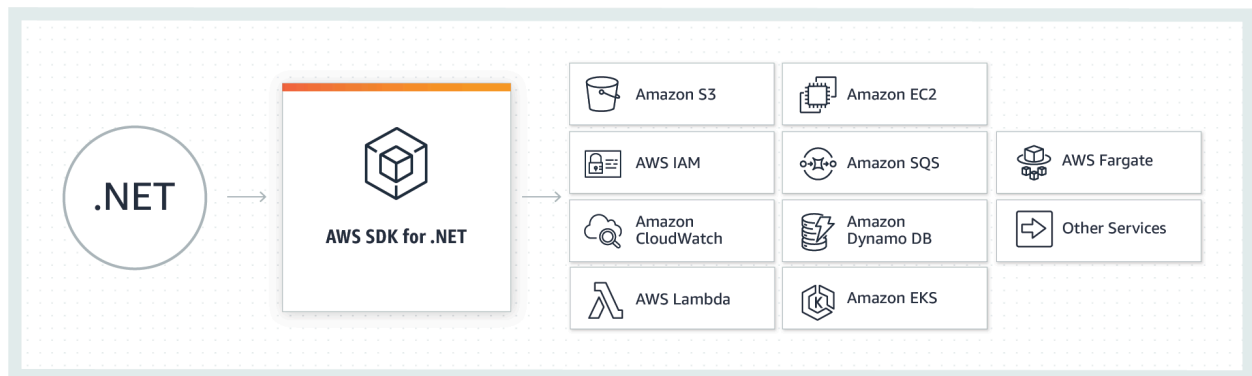


# AWS SDK (Software Development Kit)

The goal of AWS SDK (Software Development Kit) is to take the complexity of developing applications which integrate with AWS by providing language specific APIs for AWS services. As of 2024 the AWS SDK is shipped in 10 different languages: JavaScript, Python, PHP, .NET, Ruby, Java, Go, Node.js, C++ and SAP ABAP<sup>31</sup>.

Moreover, for each language that has a supported SDK we have a developer guide and an API reference guide. For example you can check out the PHP developer guide<sup>32</sup> and the .NET developer guide<sup>33</sup>. Also, regarding the API reference guide we can check out as an example the one targeting Ruby<sup>34</sup> and the one targeting JavaScript<sup>35</sup>.

Lastly, we can summarize that an SDK provides a set of libraries which allows programmers to access AWS services from code - as shown in the illustration below for .NET as an example<sup>36</sup>.



<sup>31</sup> <https://aws.amazon.com/developer/tools/>

<sup>32</sup> <https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/welcome.html>

<sup>33</sup> <https://docs.aws.amazon.com/sdk-for-net/latest/developer-guide/welcome.html>

<sup>34</sup> <https://docs.aws.amazon.com/sdk-for-ruby/v3/api/>

<sup>35</sup> <https://docs.aws.amazon.com/AWSJavaScriptSDK/latest/index.html>

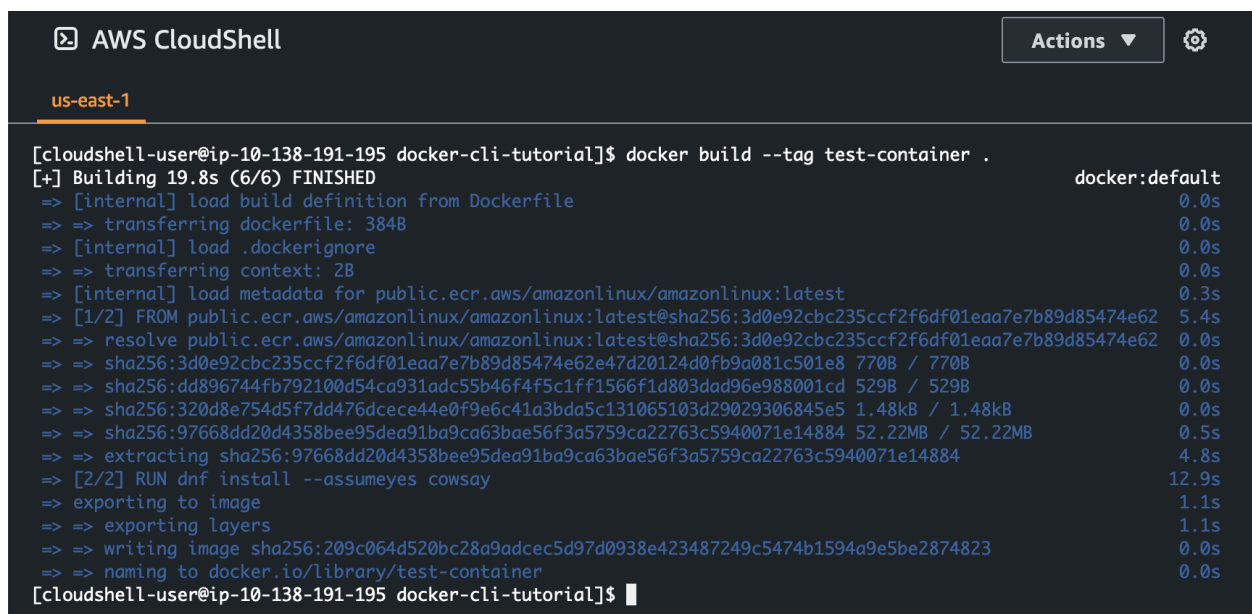
<sup>36</sup> <https://aws.amazon.com/sdk-for-net>

# AWS CloudShell

AWS CloudShell is basically a terminal in a web browser that allows exploring and managing cloud resources. It allows signing in quickly with pre-configured credentials using a pre-installed fully managed Amazon Linux 2 environment<sup>37</sup>. The compute environment has 1 vCPU and 2 GiB RAM<sup>38</sup>.

Overall, in order to start CloudShell we can search for it in the navigation bar (by typing CloudShell) or choose the “CloudShell” on the the “Console Toolbar” located on the lower left corner of the console<sup>39</sup>. Since the beginning of 2024, AWS CloudShell has built in-support for Docker. It allows spinning up containers, building, pushing them to registers and more<sup>40</sup>. An example of that is shown in the screenshot below<sup>41</sup>.

Lastly, CloudShell is based on the AWS CLI aka “AWS Command Line Interface”<sup>42</sup>. It also comes with pre-installed tools/utilities like: tar, sudo, pip, make, git, wget, vim and more. Also, AWS CloudShell supports up to 1 GB of persistent storage (in the user home folder - \$HOME) in each AWS region with no additional cost<sup>43</sup>.



```

AWS CloudShell Actions [Settings]
us-east-1
[cloudshell-user@ip-10-138-191-195 docker-cli-tutorial]$ docker build --tag test-container .
[+] Building 19.8s (6/6) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile              0.0s
=> => transferring dockerfile: 384B                             0.0s
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                    0.0s
=> [internal] load metadata for public.ecr.aws/amazonlinux/amazonlinux:latest 0.3s
=> [1/2] FROM public.ecr.aws/amazonlinux/amazonlinux:latest@sha256:3d0e92cbc235ccf2f6df01eaa7e7b89d85474e62 5.4s
=> => resolve public.ecr.aws/amazonlinux/amazonlinux:latest@sha256:3d0e92cbc235ccf2f6df01eaa7e7b89d85474e62 0.0s
=> => sha256:3d0e92cbc235ccf2f6df01eaa7e7b89d85474e62e47d20124d0fb9a081c501e8 770B / 770B 0.0s
=> => sha256:dd896744fb792100d54ca931adc55b46f4f5c1ff1566f1d803dad96e988001cd 529B / 529B 0.0s
=> => sha256:320d8e754d5f7dd476dcece44e0f9e6c41a3bda5c131065103d29029306845e5 1.48kB / 1.48kB 0.0s
=> => sha256:97668dd20d4358bee95dea91ba9ca63bae56f3a5759ca22763c5940071e14884 52.22MB / 52.22MB 0.5s
=> => extracting sha256:97668dd20d4358bee95dea91ba9ca63bae56f3a5759ca22763c5940071e14884 4.8s
=> [2/2] RUN dnf install --assumeyes cowsay                      12.9s
=> exporting to image                                           1.1s
=> => exporting layers                                          1.1s
=> => writing image sha256:209c064d520bc28a9adcec5d97d0938e423487249c5474b1594a9e5be2874823 0.0s
=> => naming to docker.io/library/test-container                0.0s
[cloudshell-user@ip-10-138-191-195 docker-cli-tutorial]$
```

<sup>37</sup> <https://aws.amazon.com/cloudshell/>

<sup>38</sup> <https://tutorialsdojo.com/aws-cloudshell/>

<sup>39</sup> <https://docs.aws.amazon.com/cloudshell/latest/userguide/welcome.html>

<sup>40</sup> <https://aws.amazon.com/about-aws/whats-new/2024/01/aws-cloudshell-docker-13-regions/>

<sup>41</sup> <https://www.infoq.com/news/2024/01/docker-aws-cloudshell/>

<sup>42</sup> <https://medium.com/@boutnaru/the-aws-concept-journey-aws-cli-command-line-interface-b5659962025f>

<sup>43</sup> <https://docs.aws.amazon.com/pdfs/cloudshell/latest/userguide/awsccloudshell.pdf>