aws

Administrator Guide

Amazon DCV Connection Gateway

Amazon DCV Connection Gateway: Administrator Guide

Table of Contents

What is Amazon DCV Connection Gateway?

> **Note**
>
> Amazon DCV was previously known as NICE DCV.

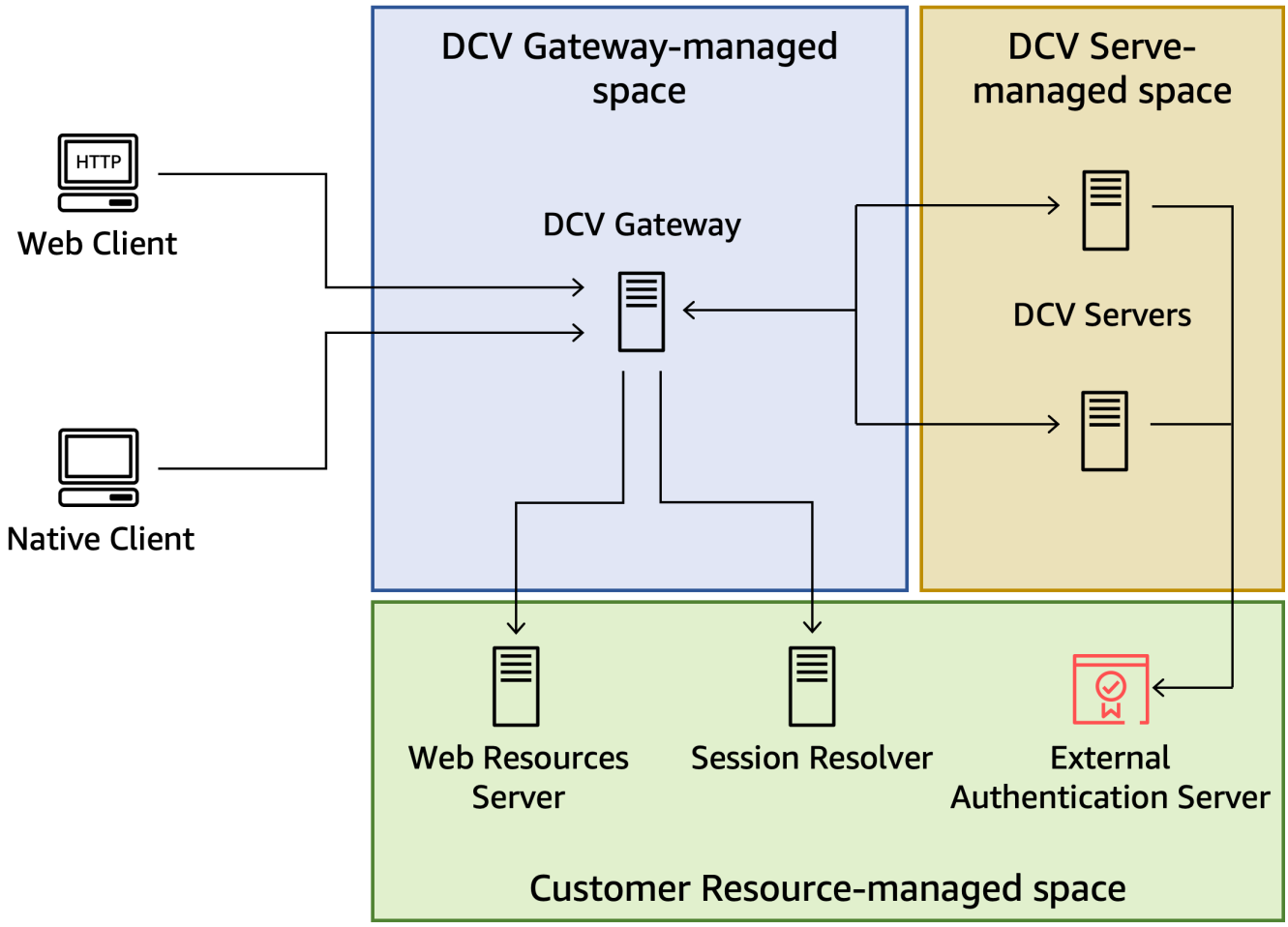
The Amazon DCV Connection Gateway is an installable software package that enables users to access a fleet of Amazon DCV servers through a single access point to a LAN or VPC. This access point is a secure and efficient platform that enables seamless remote access to virtual desktops and applications. Centralizing access management, the Amazon DCV Connection Gateway streamlines enterprise-wide remote work capabilities while maintaining robust security controls.

This guide explains how to install and configure the Amazon DCV Connection Gateway.

Topics

How the Amazon DCV Connection Gateway works

The following diagram shows the high-level view of how the Amazon DCV Connection Gateway routes traffic to a fleet of Amazon DCV servers.



When using the Amazon DCV Connection Gateway, clients connect to the gateway rather than connecting directly to a Amazon DCV server. Clients specify a *session ID*, which uniquely identifies the server they want to connect to. The Connection Gateway in turn consults a *Session Resolver* to map the session ID received by the client to a specific server and then forwards the connection to the correct destination.

Customers can define how session IDs map to their resources by implementing their Session Resolver API end-point. Customers using the Amazon DCV Session Manager can leverage its built-in session resolver.

The Amazon DCV Connection Gateway can also forward HTTP requests to a web server. This feature allows the customer to host the Amazon DCV Web Client or a custom Web application based on the Amazon DCV Web Client SDK on a dedicated web server. When a browser connects to the Connection Gateway, its request to retrieve the web page of the Amazon DCV Web Client is forwarded to the *Web Resources Server* configured in the Connection Gateway; once the browser

has retrieved and displayed that page, the Web Client will connect again to the Connection Gateway to connect to the Amazon DCV session and the Connection Gateway will forward that connection to the corresponding Amazon DCV server.

Limitations

The Amazon DCV Connection Gateway requires a Amazon DCV version greater than or equal to 2021.2 if you want to enable support for QUIC.

The Amazon DCV Connection Gateway requires that Amazon DCV is configured to use the External Authentication.

Pricing

The Amazon DCV Gateway is available at no cost for customers who are using Amazon DCV.

System requirements

For Amazon DCV Connection Gateway to run properly, your system must meet the following requirements.

Operating system	• Amazon Linux 2 • RHEL/CentOS 9 • Ubuntu 20.04 • Ubuntu 22.04 • Ubuntu 24.04
Architecture	• 64-bit x86 • 64-bit ARM

Amazon DCV Connection Gateway network requirements

Amazon DCV Connection Gateway is usually installed on dedicated hosts, separate from Amazon DCV server machines. As depicted in the high-level overview, the Connection Gateway must have

network connectivity with the other components: the Clients, the Amazon DCV server hosts, the Session Resolver, and the Web Resources Server.

> **Note**
>
> Depending on how the machines and network are configured, the network traffic that flows to and from the different components may be bound to separate network interfaces.

Please make sure your firewall rules and security groups allow the following:

- The Connection Gateway listens for incoming connection on a TCP port specified in the configuration. This port must be reachable from the clients connecting to the gateway.
- If QUIC support is enabled, Connection Gateway listens for incoming QUIC traffic on a UDP port specified in the configuration. This port must be reachable from the clients connecting to the gateway.
- The Connection Gateway must be able to connect to Amazon DCV server hosts on the TCP port used for DCV connections, 8443 by default.
- If QUIC support is enabled, Connection Gateway must be able to connect to Amazon DCV server hosts on the UDP port used for DCV QUIC connections, 8443 by default.
- The Connection Gateway must be able to connect to the TCP port of the HTTPS end-point exposed by the Session Resolver.
- If a Web Resources Server is present, Connection Gateway must be able to connect to the TCP port of the HTTPS end-point exposed by the Web Resources Server.

If you choose to have multiple Amazon DCV Connection Gateway hosts to improve availability, then a network load balancer will be present between the clients and the Connection Gateway hosts. In this case the gateway must be reachable from the load balancer nodes. When using a load balancer you may also want to use a health-check connection; in this case the load balancer need to be able to reach the TCP port of the health-check service exposed by the Amazon DCV Connection Gateway.

If using a Network Load Balander, refer to its documentation for more details.

Setting up the Amazon DCV Connection Gateway

Setting up Amazon DCV Connection Gateway involves installing the Connection Gateway package, ensuring that it properly resolves session IDs and forwards DCV connections to the Amazon DCV server hosts.

The following topics walk you through the process of installing and setting up the Amazon DCV Connection Gateway.

Topics

Installing the Amazon DCV Connection Gateway

This section describes how to install the latest version of the Amazon DCV Connection Gateway on a Linux host. You can use multiple hosts to improve scalability and performance. For more information, see Scaling the Amazon DCV Connection Gateway.

> **Note**
>
> The Amazon DCV Connection Gateway is available for the following Linux distributions and architectures:
>
> - Amazon Linux 2 (64-bit x86 and 64-bit ARM)
> - RHEL 8.x and Rocky Linux 8.x (64-bit x86 and 64-bit ARM)
> - Ubuntu 20.04, Ubuntu 22.04, and Ubuntu 24.04 (64-bit x86 and 64-bit ARM)
>
> The following instructions are for installing the Connection Gateway on 64-bit x86 hosts. To install the Connection Gateway on 64-bit ARM hosts, for Amazon Linux, RHEL, and CentOS, replace *x86_64* with `aarch64`, and for Ubuntu, replace *amd64* with `arm64`.

To install the Connection Gateway on a Linux host

1. The Amazon DCV Connection Gateway packages are digitally signed with a secure GPG signature. To allow the package manager to verify the package signature, you must import the Amazon DCV GPG key. Run the following command to import the Amazon DCV GPG key.

   - Amazon Linux 2, RHEL, CentOS, and SUSE Linux Enterprise

     ```
     $ sudo rpm --import https://d1uj6qtbmh3dt5.cloudfront.net/NICE-GPG-KEY
     ```

   - Ubuntu

     ```
     $ wget https://d1uj6qtbmh3dt5.cloudfront.net/NICE-GPG-KEY
     ```

     ```
     $ gpg --import NICE-GPG-KEY
     ```

2. Download the Amazon DCV Connection Gateway installation package for your distribution from the Amazon DCV download website.
3. Install the package.

   - Amazon Linux 2

     ```
     $  sudo yum install -y nice-dcv-connection-gateway-2024.0.777-1.el7.x86_64.rpm
     ```

   - RHEL 8.x, and Rocky Linux 8.x

     ```
     $  sudo yum install -y nice-dcv-connection-gateway-2024.0.777-1.el8.x86_64.rpm
     ```

   - Ubuntu 20.04

     ```
     $  sudo apt install ./nice-dcv-connection-
     gateway_2024.0.777-1_amd64.ubuntu2004.deb
     ```

   - Ubuntu 22.04

     ```
     $  sudo apt install ./nice-dcv-connection-
     gateway_2024.0.777-1_amd64.ubuntu2204.deb
     ```

   - Ubuntu 24.04

```
$  sudo apt install ./nice-dcv-connection-
gateway_2024.0.777-1_amd64.ubuntu2404.deb
```

Configuring the Amazon DCV Connection Gateway

This section describes how to configure the Amazon DCV Connection Gateway. It introduces the *configuration file* used by the Connection Gateway and describes the basic configuration required to run the Connection Gateway service. For more information about all the available configuration options, see the Configuration File Reference section.

The Amazon DCV Connection Gateway configuration file is located at `/etc/dcv-connection-gateway/dcv-connection-gateway.conf`. The file uses the TOML format and is organized in sections which control different aspects of the Connection Gateway.

You can edit the configuration file using your preferred text editor.

A basic configuration file will have the following content.

```
[gateway]
web-listen-endpoints = ["0.0.0.0:8443", "[::]:8445"]
quic-listen-endpoints = ["0.0.0.0:8443"]

[resolver]
url = "https://localhost:8081"

[web-resources]
url = "https://localhost:8080"
```

Configuring the Connection Gateway Listener

The `[gateway]` section controls how the Amazon DCV Connection Gateway accepts incomig connections from the clients.

```
[gateway]
web-listen-endpoints = ["0.0.0.0:8443", "[::]:8445"]
quic-listen-endpoints = ["0.0.0.0:8443"]
...
```

This section includes two parameters: `web-listen-endpoints` and `quic-listen-endpoints` which define the list of TCP and UDP endpoints (respectively) that the Connection Gateway service will bind to and listen on. In the above example, the Connection Gateway is configured to listen for incoming TCP connections on all available IPv4 addresses on TCP port 8443, and on all available IPv6 addresses on port 8445. Also, the Connection Gateway is configured to listen for incoming UDP connections on all available IPv4 addresses on UDP port 8443. The `web-listen-endpoints` parameter is required to be set and non-empty. If the `quic-listen-endpoint` parameter is not set or empty, QUIC support is disabled.

This section also allows you to configure the certificates that Amazon DCV Connection Gateway presents to the clients:

```
[gateway]
cert-file = "/path/to/cert.pem"
cert-key-file = "/path/to/key.pem"
...
```

`cert-file` and `cert-key-file` respectively specify the path of the x.509 public certificate in PEM format and the path of the file containing the private SSL key in PKCS8 representation. If these parameters are not specified, the Connection Gateway will generate and use a *self-signed* certificate.

Configuring the Session Resolver

The `[resolver]` section controls how the Amazon DCV Connection Gateway interacts with a *Session Resolver* responsible for mapping *Session IDs* to a destination host running the Amazon DCV server

```
...
[resolver]
url = "https://localhost:8081"
...
```

This section includes a *mandatory* `url` parameter which specifies the HTTP end-point of the resolver. See Implementing a Session Resolver for more information about the implementation of this end-point.

Depending on where your session resolver end-point is located and how it authenticates connections, you may need to specify additional configuration parameters: in particular if the

end point has a certificate signed by a private Certification Authority, you may provide the corresponding `ca-file` with the path of the x.509 CA certificate in PEM format:

```
...
[resolver]
ca-file = "/path/to/resolver_ca.pem"
...
```

Or if it fits your security requirements, you can accept untrusted certificates:

```
...
[resolver]
tls-strict = false
...
```

If the session resolver HTTP end-point is configured to require mutual TLS authentication, you will also need to specify the certificate and key that the Connection Gateway uses to prove its identity to the resolver. These files can be the same as the ones specified in the `[gateway]` section.

```
...
[resolver]
cert-file = "/path/to/cert.pem"
cert-key-file = "/path/to/key.pem"
...
```

Configuring the DCV target servers

The `[dcv]` section allows to specify options used by the Amazon DCV Connection Gateway to connect to the Amazon DCV server hosts.

If you are using the Amazon DCV server with the automatically generated self-signed certificates, you can use the `tls-strict` setting to allow the Connection Gateway to connect:

```
...
[dcv]
tls-strict = false
...
```

Similarly to the `[resolver]` section, you can also use the `ca-file` setting if your fleet of DCV servers use certificates signed by a private Certificate Authority.

The `[web-resources]` section controls how the Amazon DCV Connection Gateway forwards HTTP requests to an external Web Server. In particular, the Web Server is used to host the files of a DCV Web Client, so that when a browser connects to the Connection Gateway it can retrieve the `html`, `css` and `javascript` files of the DCV Web Client.

```
...
[web-resources]
url = "https://localhost:8080"
...
```

Enabling Web Access

Configuring Web Resources

The `[web-resources]` section controls how the Amazon DCV Connection Gateway forwards HTTP requests to an external Web Server. In particular, the Web Server can be used to host the files of a DCV Web Client, so that when a browser connects to the Connection Gateway it can retrieve the `html`, `css` and `javascript` files of the DCV Web Client. By default, the DCV Connection Gateway package does not include the necessary web resources to support browser-based connections. If you would like to enable browser-based connections to your DCV server fleet, follow the instructions below.

The DCV server package contains the web resources for the DCV Web Client. To obtain these resources, you will need to download the latest DCV server package and extract the web-viewer package. Once extracted, you may host the web resources on any web server that is reachable from the DCV Connection Gateway. The following sections provide two examples, one hosting the files on a cloud-native service, the other configuring a local web server on the gateway.

Using Centralized Web Resources

The following walk through will guide you on how to host the resources on the Simple Storage Service(S3) and deliver them with Amazon CloudFront. This option is the cloud-native, centralized approach.

Prerequisites

To perform the steps below, you will need the following:

- A provisioned S3 Bucket and AWS Identity and Access Management permissions to configure it.

> **Note**
>
> If you do not have a bucket, instructions can be found here.

- IAM permissions to use CloudShell.
- IAM permissions to create and configure a CloudFront distribution.

Hosting Web Resources

1. Open a CloudShell terminal.
2. Create a temporary directory to store your download by running the following command:

```
$ mkdir /tmp/dcvgw/
```

3. Download the DCV Server:

```
$ wget https://d1uj6qtbmh3dt5.cloudfront.net/nice-dcv-el7-aarch64.tgz
```

4. Extract your download to your temporary directory and rename it:

```
$ tar -xvzf nice-dcv-el7-aarch64.tgz -C /tmp/dcvgw/
mv /tmp/dcvgw/nice-dcv* /tmp/dcvgw/dcv-server-packages
```

5. Unpack the rpm to gain access to the web resources:

```
$ rpm2cpio /tmp/dcvgw/dcv-server-packages/nice-dcv-web-viewer*.rpm | cpio -idmv
```

6. Upload the assets to your S3 bucket:

```
$ aws s3 cp /tmp/dcvgw/dcv-server-packages/usr/share/dcv/www/ s3://BUCKET-NAME/ --
recursive
```

Delivering Web Resources

To keep your S3 bucket protected from the public internet, you will need to create a CloudFront distribution to deliver the web resources. As a best practice, you should use origin access control (OAC) to configure restricted CloudFront access to your bucket. To read more about OAC, see this documentation.

1. Navigate to the CloudFront console.
2. Choose **Create distribution**.
3. For the **Origin domain** drop down menu, choose your S3 bucket that will host the web resources.
4. For **Origin access**, choose **Origin access control settings (recommended)**.
   a. This will populate a new section called **Origin access control**. Select **Create control setting**.
   b. Keep the default selections and choose **Create**.
   c. Choose **Create distribution** at the bottom of the page.
   d. Creating the distribution will create a banner at the top that reads “The S3 bucket policy needs to be updated”. Within the banner, choose the **Copy policy** button and paste the policy locally.
   e. Take note of your **Distribution domain name** within the **Details** section of your distribution.
   f. Navigate to your S3 bucket within the S3 console.
   g. Within your bucket, navigate to the **Permissions** tab.
   h. Within the **Bucket policy** section, select **Edit**.
   i. Paste the policy that you acquired from the banner button within the policy editor.
   j. Choose **Save changes**.

Now that your web resources are being hosted in S3 and delivered from CloudFront, you need to point your DCV Connection Gateway to your distribution so that it can serve the DCV static assets when users initiate browser-based connections. This can be done by adding the attribute below to the [web-resources] section of your gateway’s configuration file.

```
[web-resources]
url = DistributionDomainName
```

Once you have modified the configuration, reload the gateway.

Using Local Web Resources

The following walk through will guide you on how to host the resources locally on the gateway. Note that since each gateway is hosting their own web resources, if you ever need to update the resources, you will need to do so across your gateway fleet. The instructions below will target packages for ARM-based Amazon Linux 2 instances. If you have leveraged a different distribution

for your DCV Connection Gateway, you will need to replace the URL in step three with your respective distribution. This can be retrieved from the Amazon DCV downloads page under Amazon DCV Server. If you need to update the web resources with this approach, since the resources are local to the machine, you will need to either update your Amazon Machine Image (AMI) or push an update through a remote administration tool, such as AWS Systems Manager.

Locally Hosting Web Resources

1. SSH into your DCV Connection Gateway.
2. Create a temporary directory to hold your download by running the following command:

```
$ mkdir /tmp/dcvgw/
```

3. Download the latest version of DCV Server:

> **Note**
>
> If you are running an ARM-based Amazon Linux 2 instance:
>
> ```
> $ wget https://d1uj6qtbmh3dt5.cloudfront.net/nice-dcv-el7-aarch64.tgz
> ```

4. Extract your download to your temporary directory and rename it:

```
$ tar -xvzf nice-dcv-el7-aarch64.tgz -C /tmp/dcvgw/
mv /tmp/dcvgw/nice-dcv* /tmp/dcvgw/dcv-server-packages
```

5. Install the web resources package:

```
$ sudo yum localinstall -y /tmp/dcvgw/dcv-server-packages/nice-dcv-web-viewer*.rpm
```

6. Open your DCV Connection Gateway configuration file in your preferred text editor:

```
$ sudo vi /etc/dcv-connection-gateway/dcv-connection-gateway.conf
```

7. Within your `[web-resources]` section, add the following line:

```
$ local-resources-path = "/usr/share/dcv/www"
```

8. If your Amazon DCV Connection Gateway service is already running, restart it with the following command:

```
$ sudo systemctl restart dcv-connection-gateway
```

9. If your DCV Connection Gateway service is stopped, start it.

Optional Security Settings

> **Note**
>
> If you are not interested in using the DCV Web Client or if client machines retrieve the DCV Web Client from a separate server, you can skip this section.

If the url parameter is specified, it points to the HTTP end-point of a Web Server which can serve static files, in particular the `html`, `css` and `javascript` files of the DCV Web Client.

Similarly to the `[resolver]` section, you can also use the `ca-file` or the `tls-strict` settings to be able to connect to a Web server that has a certificate signed by a private Certificate Authority or a self-signed certificate.

```
...
[web-resources]
ca-file = "/path/to/resolver_ca.pem"...
```

Setting up a Session Resolver

The *Session Resolver* is the component responsible for mapping *Session IDs* to a destination host running the Amazon DCV server. The logic of this mapping is specific to how each customer designs and plans to use its infrastructure.

The following topics describe how customers can implement a *Session Resolver* that matches their requirements and configure it in the Amazon DCV Connection Gateway. Customers using the Amazon DCV Session Manager can refer to Integrating Connection Gateway with Session Manager to learn how to use the Session Resolver end-point included in the Amazon DCV Session Manager.

Topics

- Implementing a Session Resolver
- Configuration

Implementing a Session Resolver

Your session resolver service can run on the same host as the Amazon DCV Connection Gateway or it can run on a separate host. The authentication service must listen for HTTP(S) POST requests from the Connection Gateway.

The following shows the POST request format used by the Connection Gateway.

```
POST /resolveSession?
sessionId=session_id&transport=transport&clientIpAddress=clientIpAddress HTTP/1.1
accept: application/json
```

The `sessionId` parameter contains a string which uniquely identifies a DCV session, the `transport` parameter will either be `HTTP` or `QUIC`, the `clientIpAddress` will be the ip address of the client, or the load balancer ip address if the gateway is fronted by a load balancer, the `clientIpAddress` can either be an IPv4 or IPv6 address. In case the gateway cannot get the client ip, it will not be present in the request.

Your session resolver service is responsible for determining the destination host, if any, where to forward the connection and returns its response to the Connection Gateway.

- If a destination is not found, the session resolver service returns an HTTP status `404`
- If a destination is successfully identified, the session resolver service returns an HTTP status `200` and the response body must contain the following JSON:

```
{
    "SessionId": session_id,
    "TransportProtocol": transport_protocol,
    "DcvServerEndpoint": dns_name,
    "Port": port,
    "WebUrlPath": web_url_path
}
```

The `SessionId` field normally would just return the same ID that was provided as input, however, if it is useful for your use case, you can also use this field to map a client-facing session ID to a different session ID used internally by your infrastructure. The `TransportProtocol` field must be either `HTTP` or `QUIC` (uppercase).

Example session resolver python implementation

```
from flask import Flask, request
import json

app = Flask(__name__)

dcv_sessions = {
  "session-123": {
    "SessionId": "session-123",
    "Host": "dcv123.mycompany.com",
    "HttpPort": 8443,
    "QuicPort": 8443,
    "WebUrlPath": "/"
  },
  "session-456": {
    "SessionId": "session-456",
    "Host": "dcv456.mycompany.com",
    "HttpPort": 8443,
    "QuicPort": 8443,
    "WebUrlPath": "/"
  }
}

@app.route('/resolveSession', methods=['POST'])
def resolve_session():
    session_id = request.args.get('sessionId')
    transport = request.args.get('transport')
    client_ip_address = request.args.get('clientIpAddress')

    if session_id is None:
        return "Missing sessionId parameter", 400

    if transport != "HTTP" and transport != "QUIC":
        return "Invalid transport parameter: " + transport, 400

    print("Requested sessionId: " + session_id + ", transport: " + transport + ",
 clientIpAddress: " + client_ip_address)
    dcv_session = dcv_sessions.get(session_id);
    if dcv_session is None:
        return "Session id not found", 404

    response = {
        "SessionId": dcv_session['SessionId'],
        "TransportProtocol": transport,
```

```
        "DcvServerEndpoint": dcv_session['Host'],
        "Port": dcv_session["HttpPort"] if transport == "HTTP" else
 dcv_session['QuicPort'],
        "WebUrlPath": dcv_session['WebUrlPath']
    }
    return json.dumps(response)


if __name__ == '__main__':
    app.run(port=9000, host='0.0.0.0')
```

Configuration

You must configure the Amazon DCV Connection Gateway to use the Session Resolver service.

To specify a session resolver

1. Navigate to the `/etc/dcv-connection-gateway/` folder and open the `dcv-connection-gateway.conf` with your preferred text editor.
2. Locate the `[resolver]` and set the `url` parameter to the URL of your session resolver.

   ```
   [resolver]
   url = "http://localhost:9000"
   ```

3. Save and close the file.

Managing the Connection Gateway

Effective management of the Amazon DCV Connection Gateway is essential for ensuring access to your Amazon DCV servers, as well as maintaining the overall security and integrity of the system. This section will provide detailed guidance on starting, stopping, and configuring the DCV Connection Gateway.

Topics

Starting the Connection Gateway

Manually start the Connection Gateway service using the command line.

To start the Connection Gateway service

Use the following command:

```
$ sudo systemctl start dcv-connection-gateway
```

Configure the Connection Gateway service to start automatically.

To configure the Connection Gateway service to start automatically

Use the following command:

```
$ sudo systemctl enable dcv-connection-gateway
```

Stopping the Connection Gateway

Manually stop the Connection Gateway service using the command line.

To stop the Connection Gateway service

Use the following command:

```
$ sudo systemctl stop dcv-connection-gateway
```

Checking the status of the Connection Gateway

To Check the status of the Connection Gateway service using the command line.

To check the status of the Connection Gateway

Use the following command:

```
$ sudo systemctl status dcv-connection-gateway
```

Reloading the Connection Gateway configuration

To reload the configuration of the Connection Gateway using the command line.

To reload the configuration of the Connection Gateway

Use the following command:

```
$ sudo systemctl reload dcv-connection-gateway
```

Verifying the Connection Gateway connectivity

Let's assume that the Connection Gateway host is associated with a DNS name, for instance `dcv.gateway.domain`, and it is listening on TCP port 8443 and UDP port 8443. We can use the `nc` command to test the connectivity of our gateway.

To check if the Connection Gateway is reacheable with TCP

Use the following command:

```
$ nc -vz dcv.gateway.domain 8443
```

To check if the Connection Gateway is reacheable with UDP

Use the following command:

```
$ nc -uvz dcv.gateway.domain 8443
```

Understanding Connection Gateway activity logs

The Amazon DCV Connection Gateway logs its activities to a log file. Log files are useful for monitoring the state of the Connection Gateway and can be used to troubleshoot problems. This section introduces the log file used by the Amazon DCV Connection Gateway and describes how to configure all the aspects related to logging, such as location, verbosity, size, and rotation.

By default, log files produced by the Amazon DCV Connection Gateway are located in `/var/log/dcv-connection-gateway/` folder. Logs are rotated by default. The most recent log is named `gateway.log`, while older logs are named `gateway.log.N`, where `N` is a number. A bigger number indicates an older file log.

Every line in the log files uses the following format.

```
[Timestamp] [Level] [Context]: [Message]
```

Timestamps refer to the UTC time. Log level is one of `error`, `warn`, `info`, `debug`, `trace` and it is an indication of the importance of the message. By default, `debug` and `trace` messages are not included in the logs to reduce the verbosity, but while troubleshooting it is recommended to turn them on by changing the `level` parameter in the configuration. Consult the configuration file reference for a list of parameters that affect the logging behavior.

Understanding Connection Gateway metrics

The Amazon DCV Connection Gateway is able to record and emit metrics which allow customers to monitor the performance of the Connection Gateway.

The emission of metrics is disabled by default. The Amazon DCV Connection Gateway supports emitting its metrics in a format compatible with StatsD. To enable the emission of the metrics, edit the `/etc/dcv-connection-gateway/dcv-connection-gateway.conf` and add the following:

```
[metrics-reporter-statsd]
endpoints = ["127.0.0.1:8125"]
```

Note

It is up to the customer to install a StatsD service. See Sending Metrics to Amazon CloudWatch to use Amazon CloudWatch Agent as a StatsD service.
The values of `endpoints` and `port` must match the ones used by your installation of StatsD.

List of metrics

The following table lists the metrics emitted by the Amazon DCV Connection Gateway.

Name	Unit	Description
`ClientConnectionRequestCount`	Count	The number of connection requests processed by the Connection Gateway. Each DCV connection, during the connection phase, generates a single connection request
`ClientConnectionRequestTime`	Milliseconds	The time elapsed between the establishment of a connection from the DCV client to the Connection Gateway and the reception of the first message from the DCV client by the Connection Gateway
`ClientConnectionRequestTimeoutCount`	Count	The number of times a connection request has been rejected because of timeout. In other words, if a DCV client takes too long to send the first message, the connection will be actively closed by the Connection Gateway, in

Name	Unit	Description
		order to prevent malicious slow send attacks
ClientConnectionTimeoutCount	Count	The number of times a DCV connection has been closed because of a timeout between the DCV client and the Connection Gateway
ClientFailureLoginAuthenticationFailedCount	Count	The number of times a DCV connection has been rejected by the DCV server because of the authentication
ClientFailureLoginConnectionLimitReachedCount	Count	The number of times a DCV connection has been rejected by the DCV server because the maximum number of connections has been reached
ClientFailureLoginCount	Count	The number of times a DCV connection has been rejected by the DCV server
ClientFailureLoginGenericErrorCount	Count	The number of times a DCV connection has been rejected by the DCV server because of a generic error
ClientFailureLoginInternalServerErrorCount	Count	The number of times a DCV connection has been rejected by the DCV server because of an internal error

Name	Unit	Description
ClientFailureLoginInvalidConnectionIdCount	Count	The number of times a DCV connection has been rejected by the DCV server because request contains an invalid connection identifier
ClientFailureLoginInvalidSessionIdCount	Count	The number of times a DCV connection has been rejected by the DCV server because the request contains an invalid session identifier
ClientFailureLoginProtocolErrorCount	Count	The number of times a DCV connection has been rejected by the DCV server because of a protocol error
ClientFailureLoginUnknownErrorCount	Count	The number of times a DCV connection has been rejected by the DCV server because of an unknown error
ClientNetworkIn	Bytes	The number of bytes received from the clients and forwarded to the corresponding target by the Connection Gateway
ClientNetworkOut	Bytes	The number of bytes received from the targets and forwarded to a specific client by the Connection Gateway

Name	Unit	Description
ClientRequestReceptionTime	Milliseconds	The time elapsed between the establishment of a TLS connection from a client to the Connection Gateway and the reception of the HTTP request by the Connection Gateway
ClientRequestReceptionTimeoutCount	Count	The number of TLS connections dropped due to a timeout on the reception of the HTTP request. In other words, if a client takes too long to send an HTTP request after establishing the TLS connection, the TLS connection will be actively closed by the Connection Gateway, in order to prevent malicious slow send attacks
ClientSuccessfulLoginCount	Count	The number of times a DCV connection has been successfully accepted by the DCV server
ConnectionTerminatedShutdownCount	Count	The number of connections terminated due to the shutdown of the Connection Gateway
ConnectionThrottledCount	Count	The number of times a DCV connection has been rejected by the Connection Gateway because of throttling

Name	Unit	Description
ConnectionTime	Milliseconds	The time elapsed between the establishment and the termination of a connection
CurrentConnectedClients	Count	The number of DCV clients currently connected to the Connection Gateway
CurrentNetworkConnections	Count	The number of concurrent TCP/QUIC connections active from clients to the Connection Gateway and from the Connection Gateway to targets
GatewayHttpCode4XXCount	Count	The number of HTTP responses with error codes 4XX generated by the Connection Gateway
GatewayHttpCode5XXCount	Count	The number of HTTP responses with error codes 5XX generated by the Connection Gateway
GatewayInternalErrorCount	Count	The number of errors originating from the Connection Gateway itself that prevented a request from being processed successfully
LatencyOverhead	Milliseconds	Overhead introduced by the Gateway in forwarding the DCV messages

Name	Unit	Description
NetworkConnectionRequestCount	Count	The number of client connection requests processed by the gateway since startup
SessionResolverSuccessCount	Count	The number of HTTP requests to the Session Resolver which returned successfully (status code 200)
SessionResolverNotFoundCount	Count	The number of HTTP requests to the Session Resolver which returned an error because the destination host could not be found (status code 404)
SessionResolverInvalidResponseCount	Count	The number of HTTP requests to the Session Resolver which returned an error because it failed to handle the request (any status code different from 200 or 404)
SessionResolverConnectionErrorCount	Count	The number of HTTP requests to the Session Resolver which failed because the Session Resolver could not be reached
SessionResolverResponseTime	Milliseconds	The time between when an HTTP request is sent to the Session Resolver and when the corresponding response is received

Name	Unit	Description
TargetConnectionTimeoutCount	Count	The number of times a DCV connection has been closed because of a timeout between the Connection Gateway and the target (e.g., DCV server)
TargetHttpCode2xxCount	Count	The number of HTTP responses with codes 2XX generated by targets
TargetHttpCode3xxCount	Count	The number of HTTP responses with error codes 3XX generated by targets
TargetHttpCode4xxCount	Count	The number of HTTP responses with error codes 4XX generated by targets
TargetHttpCode5xxCount	Count	The number of HTTP responses with error codes 5XX generated by targets
TargetHttpResponseTime	Milliseconds	The elapsed time between the forwarding of a HTTP request to a target and the reception of the response from the target
TargetNetworkConnectionErrorCount	Count	The number of errors while enstablishing a TCP/QUIC connection to the target from the Connection Gateway

Name	Unit	Description
`TargetTlsNegotiationErrorCount`	Count	The number of TLS connection attempts initiated by the Connection Gateway that did not establish a connection with the target. Possible causes include a mismatch of ciphers or protocols
`TargetUnreachableErrorCount`	Count	The number of connection attempts initiated by the Connection Gateway that did not establish a connection with the target because the target is not reachable

Each metric specifies additional *dimensions*, which allow to filter and aggreagate the values. In particular, the Amazon DCV Connection Gateway adds a `protocol` dimension which can be set to `HTTP`, `WebSocket`, or `QUIC`, which respectively identify whether the value is related to a HTTP request, to a DCV connection using WebSockets, or to a DCV connection using QUIC.

Metrics of connection stats

The following table lists the metrics emitted by enabling the `enable-quic-connections-stats` and `enable-tcp-connections-stats` configuration parameters in the DCV and Gateway sections.

Name	Config parameter	Unit	Description
`ClientCongestionEvents`	[dcv] enable-quic-connections-stats	Count	The cumulative number of congestion events of the QUIC connection between the Connection Gateway and the

Name	Config parameter	Unit	Description
			target (e.g. DCV server)
`ClientCongestionWindow`	[dcv] enable-quic-connections-stats	Bytes	The size of the congestion window. The congestion controller determines this dynamically based on estimated bandwidth between the Connection Gateway and the target (e.g. DCV server)
`ClientDeliveryRateAppLimited`	[dcv] enable-tcp-connections-stats	Boolean	Indicates if the goodput was measured when the socket's throughput was limited by the sending application in the connection between the Connection Gateway and the target (e.g. DCV server)
`ClientRtt`	[dcv] enable-quic-connections-stats [DCV] enable-tcp-connections-stats	Milliseconds	The round trip time of the TCP or QUIC connection between the DCV client and the Connection Gateway

Name	Config parameter	Unit	Description
`ClientSegmentsLossRate`	[dcv] enable-tcp-connections-stats	Percentage	The percentage of packet loss in the TCP connection between the Connection Gateway and the target (e.g. DCV server)
`ClientSegmentsRetransRate`	[dcv] enable-tcp-connections-stats	Percentage	The percentage of packets retransmitted in the TCP connection between the Connection Gateway and the target (e.g. DCV server)
`TargetCongestionEvents`	[gateway] enable-quic-connections-stats	Count	The number of congestion events of the QUIC connection between the DCV client and the Connection Gateway
`TargetCongestionWindow`	[gateway] enable-quic-connections-stats	Bytes	The size of the congestion window. The congestion controller determines this dynamically based on estimated bandwidth between the DCV client and the Connection Gateway

Name	Config parameter	Unit	Description
`TargetDeliveryRateAppLimited`	[gateway] enable-tcp-connections-stats	Boolean	Indicates if the goodput was measured when the socket's throughput was limited by the sending application in the connection between the DCV client and the Connection Gateway
`TargetRtt`	[gateway] enable-quic-connections-stats [Gateway] enable-tcp-connections-stats	Milliseconds	The round trip time of the TCP or QUIC connection between the Connection Gateway and the target (e.g. DCV server)
`TargetSegmentsLossRate`	[gateway] enable-tcp-connections-stats	Percentage	The percentage of packet loss in the TCP connection between the DCV client and the Connection Gateway
`TargetSegmentsRetransRate`	[gateway] enable-tcp-connections-stats	Percentage	The percentage of packets retransmitted in the TCP connection between the DCV client and the Connection Gateway

Sending Metrics to Amazon CloudWatch

The Amazon CloudWatch agent can be installed on the host running the Amazon DCV Connection Gateway and can be configured to collect the metrics and send them to the CloudWatch service of your AWS account.

To send the Amazon DCV Connection Gateway metrics to Amazon CloudWatch

1. Install the Amazon CloudWatch agent on your host.

   Refer to the CloudWatch documentation for detailed instructions on how to install the agent and ensure that the required IAM roles are present.

2. Enable the stasd plugin of the Amazon CloudWatch Agent.

   Refer to the CloudWatch documentation for detailed instructions on how to enable the StatsD plugin.

3. Configure the Amazon CloudWatch Agent to collect the Amazon DCV Connection Gateway metrics.

   Create or edit the `/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json` with your preferred editor and add the following content:

```
{
    "metrics": {
        "namespace": "DCV-Connection-Gateway",
        "metrics_collected": {
            "statsd": {
                "service_address": ":8125",
                "metrics_collection_interval": 5,
                "metrics_aggregation_interval": 60
            }
        },
        "append_dimensions": {
            "InstanceId": "${aws:InstanceId}"
        }
    }
}
```

4. Restart the Amazon CloudWatch Agent.

```
sudo systemctl start amazon-cloudwatch-agent
```

5. Enable the metrics in the Amazon DCV Connection Gateway.

   Edit the `/etc/dcv-connection-gateway/dcv-connection-gateway.conf` and add the following:

   ```
   [metrics-reporter-statsd]
   endpoints = ["127.0.0.1:8125"]
   ```

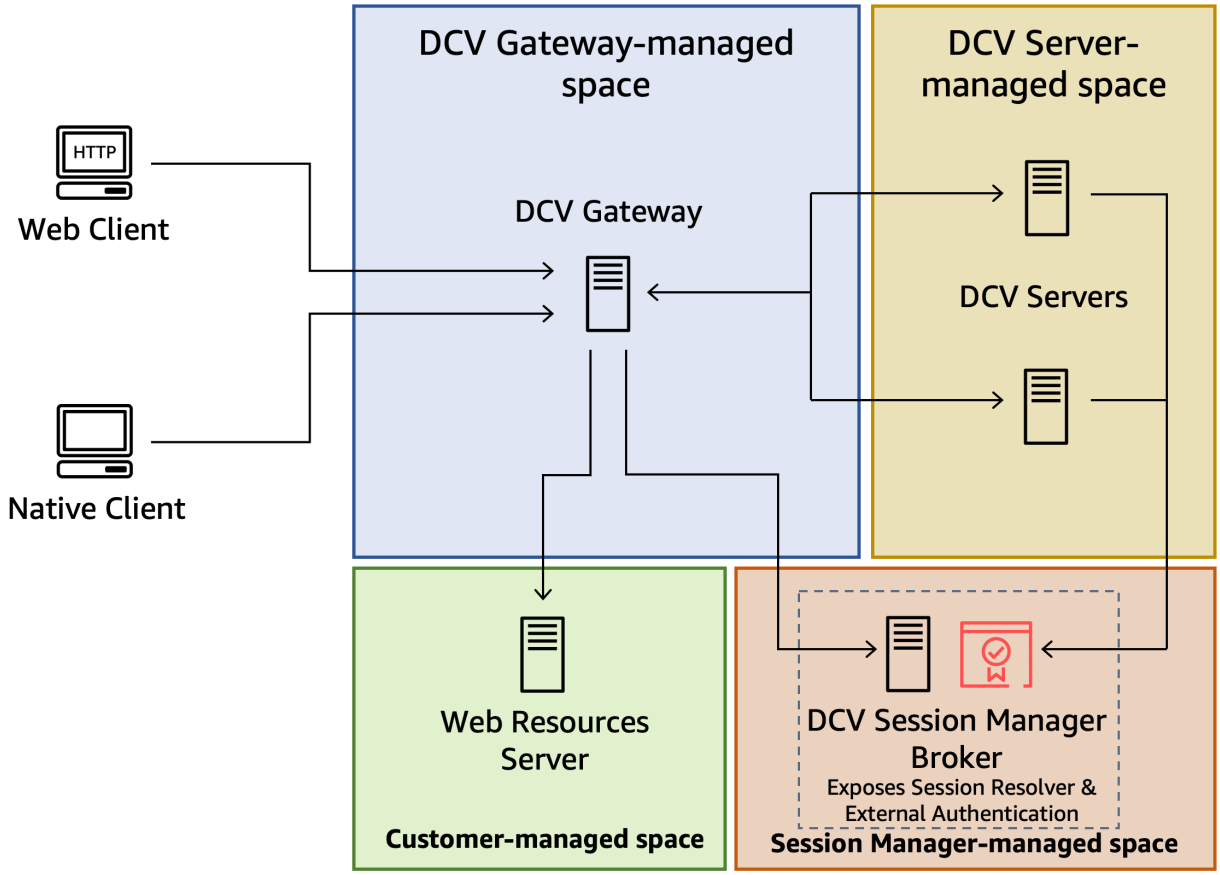
   > **Note**
   >
   > The values specified for `endpoints` and `port` must match the ones used in the `service_address` parameter of the Amazon CloudWatch Agent statsd configuration file.

6. Restart the Amazon DCV Connection Gateway service.

   ```
   sudo systemctl restart dcv-connection-gateway
   ```

Integrating Connection Gateway with Session Manager

Amazon DCV Connection Gateway can be used in conjunction with Amazon DCV Session Manager, which manages Amazon DCV server hosts and provides a Session Resolver end-point. The simplified high-level overview becomes:



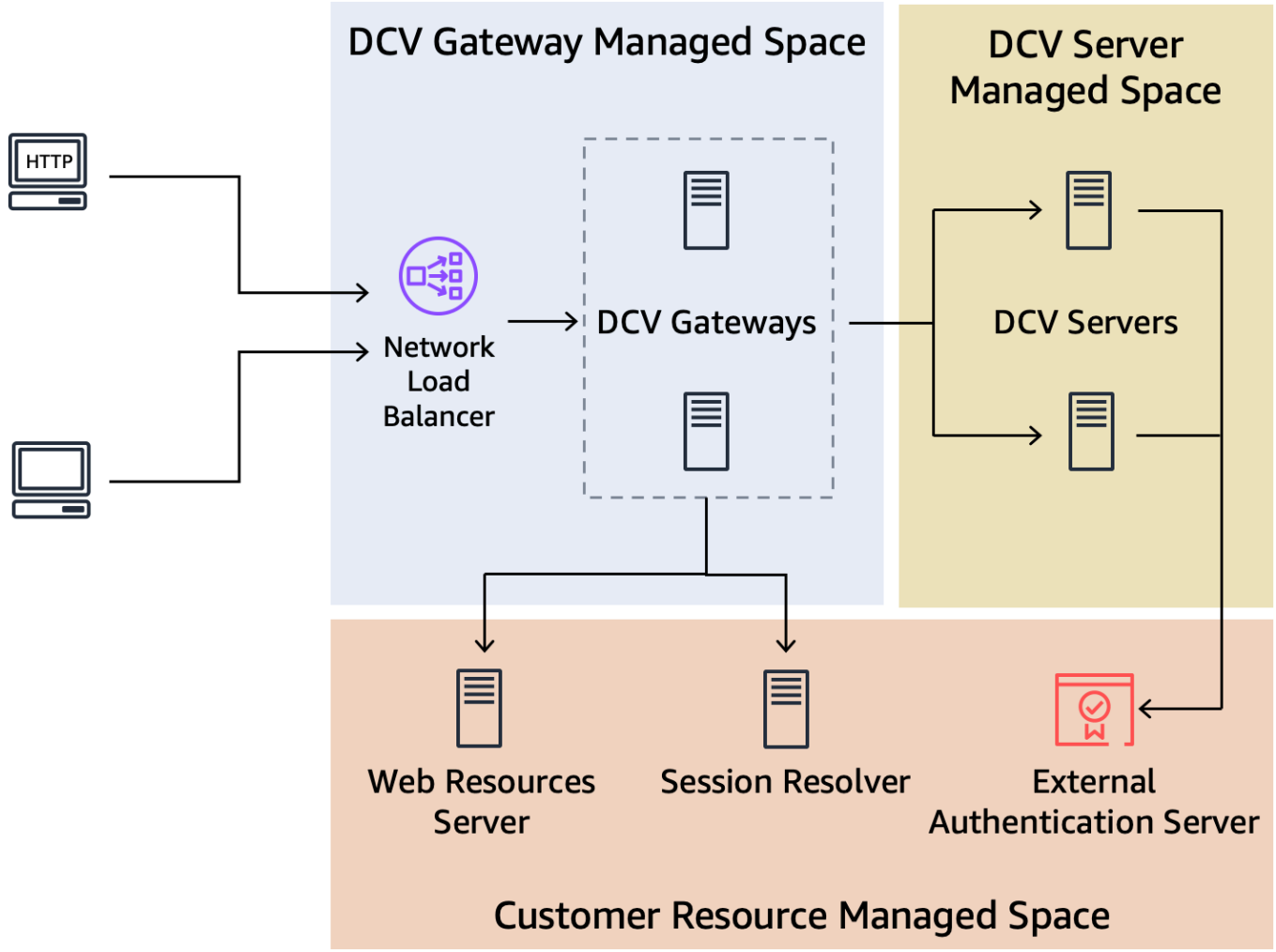
Refer to the Amazon DCV Session Manager documentation for more information about configuring the Session Resolver in Amazon DCV Session Manager.

Scaling the Amazon DCV Connection Gateway

The following topics describe how to scale Amazon DCV Connection Gateway using a fleet of gateway hosts and a Network Load Balancer.

Topics

The simplified high-level overview includes a single Connection Gateway which forwards connections to a fleet of Amazon DCV server hosts. In this architecture the Connection Gateway is a single point of failure. To increase robustness and scalability, we can use a fleet of Connection Gateway hosts and front them with a Network Load Balancer, in order to preserve the ability for clients to target a single entry point to the server-side infrastructure.



With this architecture, gateway nodes can be added or removed according to the system load without any disruption for the clients.

The Network Load Balancer can *check the health* of each instance of the Connection Gateway and uses this information to select whether one of the Connection Gateway should or should not be used to handle incoming connections.

Reporting the Health of the Connection Gateway

The Amazon DCV Connection Gateway can be configured to listen on an additional TCP port that will be used to check the health of the Connection Gateway service.

To enable the health check service in the Amazon DCV Connection Gateway, edit the `/etc/dcv-connection-gateway/dcv-connection-gateway.conf` and add the following:

```
[health-check]
bind-addr = "::"
port = 8989
```

The `bind-addr` and `port` are the IP address and TCP port used by the health check service. They need to be reachable from the Network Load Balancer. `bind-addr` can use IPv4 or IPv6 addresses.

Configuring a Network Load Balancer

The following steps summarize how to create a Network Load Balancer and highlight the settings which are needed to use a Network Load Balancer with Amazon DCV Connection Gateway. See the Network Load Balancer documentation for more detailed information.

To create a Network Load Balancer for a fleet of Amazon DCV Connection Gateway hosts

1. Navigate to the EC2 Console, select **Load Balancer** from the navigation pane and then then choose **Create Load Balancer**. For load balancer type, choose **Network Load Balancer**.
2. For **Basic Configuration** assign a **Name**, set **Scheme** to **internet-facing**, and set **Ip address type** to **IPv4**.
3. For **Network mapping** select your **VPC** and then select all the availability zones and subnets in that VPC. Make sure that your DCV Connection Gateway instances security groups allow traffic from the selected subnets.
4. For **Listeners and routing** create a TCP target group, specifying the `web-port` of the Amazon DCV Connection Gateway configuration as the port.

   For the *health check*, make sure TCP is used and override the TCP port with the one specified in the `[health-check]` section of the Amazon DCV Connection Gateway configuration.

   If you also want QUIC support, create a UDP target group, specifying the `quic-port` of the Amazon DCV Connection Gateway configuration as the port.

   For the *health check* use the same values as before: make sure TCP is used and override the TCP port with the one specified in the `[health-check]` section of the Amazon DCV Connection Gateway configuration.

> **Note**
>
> When using a TLS listener on your Elastic Load Balancer, the Target Group also needs to be set to TLS.

If you have enabled QUIC, once the Network Load Balancer is created, select it from the list, select the *UDP listener* and make sure the **Stickiness** check box is active.

Configuration File Reference

This section provides a reference for all the parameters that can be specified in the Connection Gateway configuration file. For an introduction to the configuration of Amazon DCV Connection Gateway, see Configuring the Amazon DCV Connection Gateway.

The Amazon DCV Connection Gateway configuration file is located at `/etc/dcv-connection-gateway/dcv-connection-gateway.conf`. The file uses the TOML format and is organized in sections which control different aspects of the Connection Gateway

You can edit the configuration file using your preferred text editor.

> **Note**
>
> Some of the configuration parameters can be reloaded while the gateway is running without causing disruptions for the existing connections. Others parameters instead require a restart of the service. This is denoted by the `Requires Restart` column in the table below.

Topics

[gateway] section

Parameter n	Required	Default value	Requires Restart	Description
bind-addr	Yes		Yes	This setting is **deprecated**, use web-listen-endpoints and quic-listen-endpoints instead. The socket address the gateway will be listening on for incoming DCV client connections. The value must be a valid IP address syntax.
cert-file	No		No	The path to a PEM file containing the certificate to be used by the gateway. If not specified, the Connection Gateway will use generate self-signed certificates. When this parameter is specified, cert-key-file must be used as well.
cert-key-file	No		No	The path to the private key file of the certificate. When this parameter is specified, cert-file must be used as well.
ciphers-tls	No	["TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384", "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",	No	The TLS ciphers used for the TLS communication with the clients.

Parameter n	Required	Default value	Requires Restart	Description
		`"TLS13_CHACHA20_POLY1305_SHA256", "TLS13_AES_256_GCM_SHA384", "TLS13_AES_128_GCM_SHA256"]`		
`enable-quic-connections-stats`	No	true	Yes	Whether or not to enable UDP metrics emission for the connection between DCV client and the Connection Gateway every 60 seconds. See Metrics of connection stats
`enable-tcp-connections-stats`	No	true	Yes	Whether or not to enable TCP metrics emission for the connection between DCV client and the Connection Gateway every 60 seconds. See Metrics of connection stats
`graceful-shutdown-timeout`	No	10	Yes	When receiving a shutdown signal, the Connection Gateway waits for the specified number of seconds before closing all connections and exiting.
`minimum-tls-version`	No	`"tls12"`	No	The minimum TLS version used for the TLS communication with the clients. The value can be `"tls12"` or `"tls13"`.

Parameter n	Required	Default value	Requires Restart	Description
`quic-idle-timeout`	No	10	Yes	The timeout in seconds after which an inactive QUIC connection with a client is closed by the Connection Gateway.
`quic-listen-endpoints`	No	[]	Yes	The list of endpoints the gateway will be listening on for incoming UDP connections from DCV clients. An endpoint is defined as a *`ip-address`* `[:`*`port`*`]` pair, where *`ip-address`* is a valid IPv4 or IPv6 address and *`port`* is a UDP port. The *`port`* field in the endpoint is optional, and if not specified the `quic-port` parameter will be assumed as port. If this parameter is not set or set to an empty list, QUIC support will be disabled.
`quic-max-connections`	No	1000	Yes	The maximum number of concurrent QUIC connections the Connection Gateway is going to accept. After that limit, a new incoming connection will be rejected.
`quic-port`	No	8443	Yes	The default UDP port that will be associated to an endpoint without the `port` field in `quic-listen-endpoints` .
`tcp-idle-timeout`	No	10	Yes	The timeout in seconds after which an inactive TCP connection with a client is closed by the Connection Gateway.

Parameter n	Required	Default value	Requires Restart	Description
`tcp-max-connections`	No	1000	Yes	The maximum number of concurrent TCP connections the Connection Gateway is going to accept. After that limit, a new incoming connection will be rejected.
`web-listen-endpoints`	Yes		Yes	The list of endpoints the gateway will be listening on for incoming WebSocket and HTTP connections from DCV clients. An endpoint is defined as a *`ip-address`* `[:`*`port`*`]` pair, where *`ip-address`* is a valid IPv4 or IPv6 address and *`port`* is a TCP port. The *`port`* field in the endpoint is optional, and if not specified the `web-port` parameter will be assumed as port.
`web-port`	No	8443	Yes	The default TCP port that will be associated to an endpoint without the `port` field in `web-listen-endpoints` .

[log] section

Parameter n	Required	Default value	Requires Restart	Description
`directory`	No	`/var/log/dcv-conne`	Yes	The directory where gateway log files are going to be written.

Parameter n	Required	Default value	Requires Restart	Description
		ction-gateway		
level	No	info	No	The log level verbosity. Possible values are sorted by increasing verbosity: error, warning, info, debug, trace.
max-file-size	No	10485760	Yes	When a log file size reaches the specfied size in bytes, it will be rotated. A new log file will be created and further log events will be placed in the new file.
rolling-frequency	No	every-day	Yes	The temporal frequency with which log files will be rotated. Valid values are: every-day , every-hour , every-minute .
rotate	No	9	Yes	The maximum number of log files preserved in the rotation. Each time a rotation happens and this number is reached, the oldest log file will be deleted.

[health-check] section

Parameter n	Required	Default value	Requires Restart	Description
bind-addr	No		Yes	The socket address the gateway will be listening on for incoming health check requests. The value must be a valid IP address syntax. If this parameter is not

Parameter n	Required	Default value	Requires Restart	Description
				specified, the health check service will be disabled.
port	No	8888	Yes	The TCP port the gateway will be listening on for incoming health check requests. The value must be a valid port number.

[dcv] section

Parameter n	Required	Default value	Requires Restart	Description
ca-file	No		No	If this setting is active, the certificates presented by the DCV servers will be validated only against the Certificate-Authority's certificate specified in this file.
ciphers-tls	No	["TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384", "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256", "TLS13_CHACHA20_POLY1305_SHA256",	No	The TLS ciphers used for the TLS communication with the Amazon DCV server hosts.

Parameter n	Required	Default value	Requires Restart	Description
		"TLS13_AES_256_GCM_SHA384", "TLS13_AES_128_GCM_SHA256"]		
enable-quic-connections-stats	No	true	Yes	Whether or not to enable UDP metrics emission for the connection between Connection Gateway and the Amazon DCV server every 60 seconds. See Metrics of connection stats
enable-tcp-connections-stats	No	true	Yes	Whether or not to enable TCP metrics emission for the connection between Connection Gateway and the Amazon DCV server every 60 seconds. See Metrics of connection stats
minimum-tls-version	No	"tls12"	No	The minimum TLS version used for the TLS communication with the Amazon DCV server hosts. The value can be "tls12" or "tls13".
tls-strict	No	true	No	Whether to enable or not the verification against a trusted Certificate-Authority for the certificate presented by the Amazon DCV server. The value can be true or false.

[resolver] section

Parameter n	Required	Default value	Requires Restart	Description
ca-file	No		No	If this setting is active, the certificates presented by the resolver will be validated only against the Certificate-Authority's certificate specified in this file.
cert-file	No		No	The path to a PEM file containing the certificate the gateway will present to the Session Resolver end-point. This setting is required if the Session Manager requires mutual TLS authentication. When this parameter is specified, cert-key-file must be used as well.
cert-key-file	No		No	The path to the private key file of the certificate. When this parameter is specified, cert-file must be used as well.
ciphers-tls	No	["TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384", "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256", "TLS13_CHACHA20_PO	No	The TLS ciphers used for the TLS communication with the Session Resolver.

Parameter n	Required	Default value	Requires Restart	Description
		LY1305_SHA256", "TLS13_AES_256_GCM_SHA384", "TLS13_AES_128_GCM_SHA256"]		
minimum-tls-version	No	"tls12"	No	The minimum TLS version used for the TLS communication with the resolver. The value can be "tls12" or "tls13".
http-establish-timeout	No	10	No	The timeout in seconds used when establishing connections with the resolver.
tls-strict	No	true	No	Whether to enable or not the verification against a trusted Certificate-Authority for the certificate presented by the Session Resolver. The value can be true or false.
url	Yes		No	The url of the Session Resolver. The url host must be a domain name, ip addresses are not supported.

[web-resources] section

Parameter n	Required	Default value	Requires Restart	Description
ca-file	No		No	If this setting is active, the certificates presented by the web resources server will be validated only against the Certificate-Authority's certificate specified in this file.
ciphers-tls	No	["TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384", "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256", "TLS13_CHACHA20_POLY1305_SHA256", "TLS13_AES_256_GCM_SHA384", "TLS13_AES_128_GCM_SHA256"]	No	The TLS ciphers used for the TLS communication with the Web Resources server.
local-resources-	No	{ "strict-transport-	Yes	The HTTP headers that are set on the static web resources used when connecting via web-based Client.

Parameter n	Required	Default value	Requires Restart	Description
http-headers		security" = "max-age=31536000", "content-security-policy" = "upgrade-insecure-requests;", "x-content-type-options" = "nosniff", "x-frame-options" = "SAMEORIGIN" }		
local-resources-path	No		Yes	Local path where the DCV web resources are stored. Web-based DCV connections will be served these resources.

Parameter n	Required	Default value	Requires Restart	Description
`minimum-tls-version`	No	`"tls12"`	No	The minimum TLS version used for the TLS communication with the Web Resources Server. The value can be `"tls12"` or `"tls13"`.
`http-establish-timeout`	No	10	No	The timeout in seconds used when establishing HTTP connections with the Web Resources server.
`tls-strict`	No	true	No	Whether to enable or not the verification against a trusted Certificate-Authority for the certificate presented by the Web Resources server. The value can be `true` or `false`.
`url`	No		No	The url of the Web Resources Server. The url host must be a domain name, ip addresses are not supported. If not specified, the gateway will not forward requests for static web resources.

`[metrics-reporter-statsd]` section

Parameter n	Required	Default value	Requires Restart	Description
`endpoints`	No		Yes	The IP where the statsd service is located and metrics can be pushed to. If this parameter is not specified, the StatsD metric reporter will be disabled. Syntax as ["IP:Port"].

Parameter n	Required	Default value	Requires Restart	Description
`port`	No	8125	Yes	The UDP port of the statsd service.

Release notes and document history for Amazon DCV Connection Gateway

This page provides the release notes and document history for Amazon DCV Connection Gateway.

Topics

Amazon DCV Connection Gateway release notes

This section provides an overview of the major updates, feature releases, and bug fixes for Amazon DCV Connection Gateway. All the updates are organized by release date. We update the documentation frequently to address the feedback that you send us.

Topics

2024.0-777— October 31, 2024

Build numbers	Changes and bug fixes	
777	• Fixed file storage and printer redirection when using the `local-resources-path` configuration setting. • Removed runtime dependency on `openssl`. • Added `TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256` to the default ciphers. • Updated WebSocket connection established timeout to 5 seconds. • Added `quic-establish-timeout` setting.	

2023.1-710— March 6, 2024

Build numbers	Changes and bug fixes	
710	• Minor bug fixes	

2023.1-705— February 26, 2024

Build numbers	Changes and bug fixes	
705	• Updated SSRF/XSS • Bug fixes and security improvements	

2023.1-692— January 29, 2024

Build numbers	Changes and bug fixes	
692	• Updated SSRF/XSS • Bug fixes and security improvements	

2023.1-671— November 9, 2023

Build numbers	Changes and bug fixes	
671	• Improved throttling mechanism to take CPU load into account • Added `enable-tcp-connections-stats` and `enable-quic-connections-stats` flags in the dcv and gateway sections in order to enable detailed connection statistics metrics on client and server side. • Bug fixes and performance improvements	

2023.0-531— March 28, 2023

Build numbers	Changes and bug fixes	
531	• Added new metrics. • Fixed a bug preventing the start of the Amazon DCV Connection Gateway on Graviton instances.	

2022.2-427— November 11, 2022

Build numbers	Changes and bug fixes	
427	• Added new metrics.	

2022.1-377— June 29, 2022

Build numbers	New features	Changes and bug fixes
377	• Added support for Ubuntu 22.04 and Rocky Linux 8.5 and higher.	• Fixed a problem preventing QUIC connections to be closed when an error occurs in the server.

2022.0-351— May 19, 2022

Build numbers	Changes and bug fixes
351	• Fixed WebSocket performance problem that could occur in case of latency between the gateway and the server.

2022.0-322— March 23, 2022

Build numbers	Changes and bug fixes
322	• Handle HTTP DELETE method for DCV resources.

2022.0-310— February 23, 2022

Build numbers	Changes and bug fixes
310	• It is now possible to configure the Amazon DCV Connection Gateway to listen on a specific network interface or on specific IPv4 or IPv6 addresses. • Leverage systemd sandboxing features when they are available. • Support session resolver URLs with a path.

2021.3-251— December 20, 2021

Build numbers	Changes and bug fixes
251	• The initial release of Amazon DCV Connectio n Gateway.

Document history

The following table describes the documentation for this release of Amazon DCV Connection Gateway.

Change	Description	Date
Release of Amazon DCV Connection Gateway 2024.0-777;	Amazon DCV Connectio n Gateway 2023.0-777 is now available. For more information, see 2024.0-777 — October 31, 2024.	October 31, 2024
Release of Amazon DCV Connection Gateway 2023.1-710;	Amazon DCV Connection Gateway 2023.1-710 is now available. For more informati	March 6, 2024

Change	Description	Date
	on, see 2023.1-710— March 6, 2024.	
Release of Amazon DCV Connection Gateway 2023.1-705;	Amazon DCV Connection Gateway 2023.1-705 is now available. For more information, see 2023.1-705 — February 26, 2024.	February 26, 2024
Release of Amazon DCV Connection Gateway 2023.1-692;	Amazon DCV Connection Gateway 2023.1-692 is now available. For more information, see 2023.1-692— January 29, 2024.	January 29, 2024
Release of Amazon DCV Connection Gateway 2023.1-671;	Amazon DCV Connection Gateway 2023.1 is now available. For more information, see 2023.1-671— November 9, 2023.	November 9, 2023
Release of Amazon DCV Connection Gateway 2023.0;	Amazon DCV Connection Gateway 2023.0 is now available. For more information, see 2023.0-531— March 28, 2023.	March 28, 2023
Release of Amazon DCV Connection Gateway 2022.2;	Amazon DCV Connection Gateway 2022.2 is now available. For more information, see 2022.2-427— November 11, 2022.	November 11, 2022

Change	Description	Date
Release of Amazon DCV Connection Gateway 2022.1;	Amazon DCV Connection Gateway 2022.1 is now available. For more information, see 2022.1-377— June 29, 2022.	June 29, 2022
Release of Amazon DCV Connection Gateway 2022.0;	Amazon DCV Connection Gateway 2022.0 is now available. For more information, see 2022.0-310— February 23, 2022.	February 23, 2022
Initial release of Amazon DCV Connection Gateway	The first publication of this content.	December 20, 2021