aws

관리자 안내서

NICE DCV 세션 관리자

NICE DCV 세션 관리자: 관리자 안내서

Table of Contents

NICE DCV 세션 관리자란 무엇인가요?

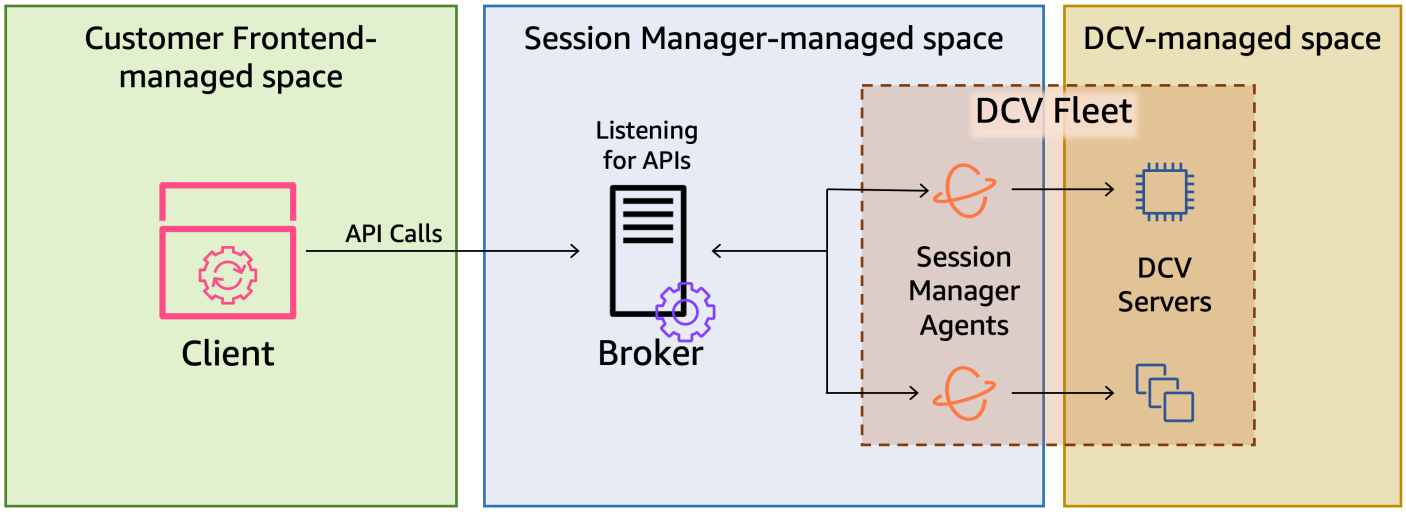
NICE DCV 세션 관리자는 설치 가능한 소프트웨어 패키지(에이전트 및 브로커)와 애플리케이션 프로그래밍 인터페이스(API)의 모음으로, 개발자와 독립 소프트웨어 개발 판매 회사(ISV)가 NICE DCV 서버 전체에서 NICE DCV 세션의 수명 주기를 프로그래밍 방식으로 생성하고 관리하는 프런트엔드 애플리케이션을 쉽게 구축할 수 있도록 합니다.

이 안내서는 세션 관리자 에이전트 및 브로커를 설치하고 구성하는 방법을 설명합니다. 세션 관리자 API 사용에 대한 자세한 내용은 NICE DCV 세션 관리자 개발자 안내서를 참조하세요.

주제

세션 관리자의 작동 방식

다음 다이어그램은 세션 관리자의 종합적 구성 요소를 보여줍니다.



브로커

브로커는 세션 관리자 API를 호스팅하고 노출하는 웹 서버입니다. 브로커는 클라이언트로부터 NICE DCV 세션을 관리하기 위한 API 요청을 수신 및 처리한 다음 관련 에이전트에 지침을 전달합니다. 브로커는 NICE DCV 서버와 분리된 호스트에 설치해야 하지만 클라이언트가 브로커에 액세스할 수 있어야 하고 브로커가 에이전트에 액세스할 수 있어야 합니다.

에이전트

에이전트는 플릿의 각 NICE DCV 서버에 설치됩니다. 에이전트는 브로커로부터 지침을 받아 해당 NICE DCV 서버에서 실행합니다. 에이전트는 또한 NICE DCV 서버의 상태를 모니터링하고 정기적으로 상태 업데이트를 브로커에 보냅니다.

API

세션 관리자는 NICE DCV 서버 플릿에서 NICE DCV 세션을 관리하는 데 사용할 수 있는 REST API(애플리케이션 프로그래밍 인터페이스) 모음을 제공합니다. API는 브로커에서 호스팅되고 브로커에 의해 노출됩니다. 개발자는 API를 호출하는 사용자 지정 세션 관리 클라이언트를 구축할 수 있습니다.

클라이언트

클라이언트는 브로커가 표시하는 세션 관리자 API를 호출하기 위해 개발하는 프런트엔드 애플리케이션 또는 포털입니다. 최종 사용자는 클라이언트를 사용하여 플릿의 NICE DCV 서버에서 호스팅되는 세션을 관리합니다.

액세스 토큰

API 요청을 하려면 액세스 토큰을 제공해야 합니다. 등록된 클라이언트 API를 통해 브로커 또는 외부 권한 부여 서버에서 토큰을 요청할 수 있습니다. 토큰을 요청하고 액세스하려면 클라이언트 API가 유효한 보안 인증 정보를 제공해야 합니다.

클라이언트 API

클라이언트 API는 Swagger Codegen을 사용하여 세션 관리자 API 정의 YAML 파일에서 생성됩니다. 클라이언트 API는 API 요청을 하는 데 사용됩니다.

NICE DCV 세션

클라이언트가 연결할 수 있는 NICE DCV 서버에 NICE DCV 세션을 만들어야 합니다. 활성 세션이 있는 경우에만 클라이언트가 NICE DCV 서버에 연결할 수 있습니다. NICE DCV는 콘솔 및 가상 세션을 지원합니다. 세션 관리자 API를 사용하여 NICE DCV 세션의 수명 주기를 관리할 수 있습니다. NICE DCV 세션은 다음 상태 중 하나일 수 있습니다.

- `CREATING`—브로커가 세션을 생성하는 중입니다.

- READY—세션이 클라이언트 연결을 수락할 준비가 되었습니다.
- DELETING—세션을 삭제 중입니다.
- DELETED—세션이 삭제되었습니다.
- UNKNOWN—세션 상태를 확인할 수 없습니다. 브로커와 에이전트가 통신하지 못할 수 있습니다.

기능

DCV 세션 관리자는 다음 기능을 제공합니다.

- NICE DCV 세션 정보 제공 - 여러 NICE DCV 서버에서 실행되는 세션에 대한 정보를 가져옵니다.
- 여러 NICE DCV 세션의 수명 주기 관리 - 한 번의 API 요청으로 여러 NICE DCV 서버에서 여러 사용자에 대한 여러 세션을 생성하거나 삭제합니다.
- 태그 지원 - 세션을 생성할 때 사용자 지정 태그를 사용하여 NICE DCV 서버 그룹을 대상으로 지정합니다.
- 여러 NICE DCV 세션에 대한 권한 관리 - 한 번의 API 요청으로 여러 세션에 대한 사용자 권한을 수정합니다.
- 연결 정보 제공 - NICE DCV 세션의 클라이언트 연결 정보를 검색합니다.
- 클라우드 및 온프레미스 지원 - AWS에서 온프레미스 또는 대체 클라우드 기반 서버와 함께 세션 관리자를 사용할 수 있습니다.

제한 사항

세션 관리자는 리소스 프로비저닝 기능을 제공하지 않습니다. Amazon EC2 인스턴스에서 NICE DCV를 실행하는 경우 인프라 규모 조정을 관리하기 위해 Amazon EC2 Auto Scaling과 같은 추가 AWS 서비스를 사용해야 할 수 있습니다.

요금

EC2 인스턴스를 실행하는 AWS 고객은 세션 관리자를 무료로 사용할 수 있습니다.

온프레미스 고객은 NICE DCV Plus 또는 DCV Professional Plus 라이선스가 필요합니다. NICE DCV Plus 또는 NICE DCV Professional Plus 라이선스를 구매하는 방법에 대한 자세한 내용은 NICE 웹사이트의 구매 방법을 참조하고 해당 리전의 NICE 유통업체 또는 리셀러를 찾아보세요. 모든 온프레미스

고객이 DCV 세션 관리자를 테스트해 볼 수 있도록 하기 위해 NICE DCV 버전 2021.0부터 라이선스 요구 사항이 적용됩니다.

자세한 내용은 NICE DCV 관리자 안내서의 NICE DCV 서버 라이선스를 참조하세요.

NICE DCV 세션 관리자 요구 사항

NICE DCV 세션 관리자 에이전트 및 브로커의 요구 사항은 다음과 같습니다.

	브로커	Agent

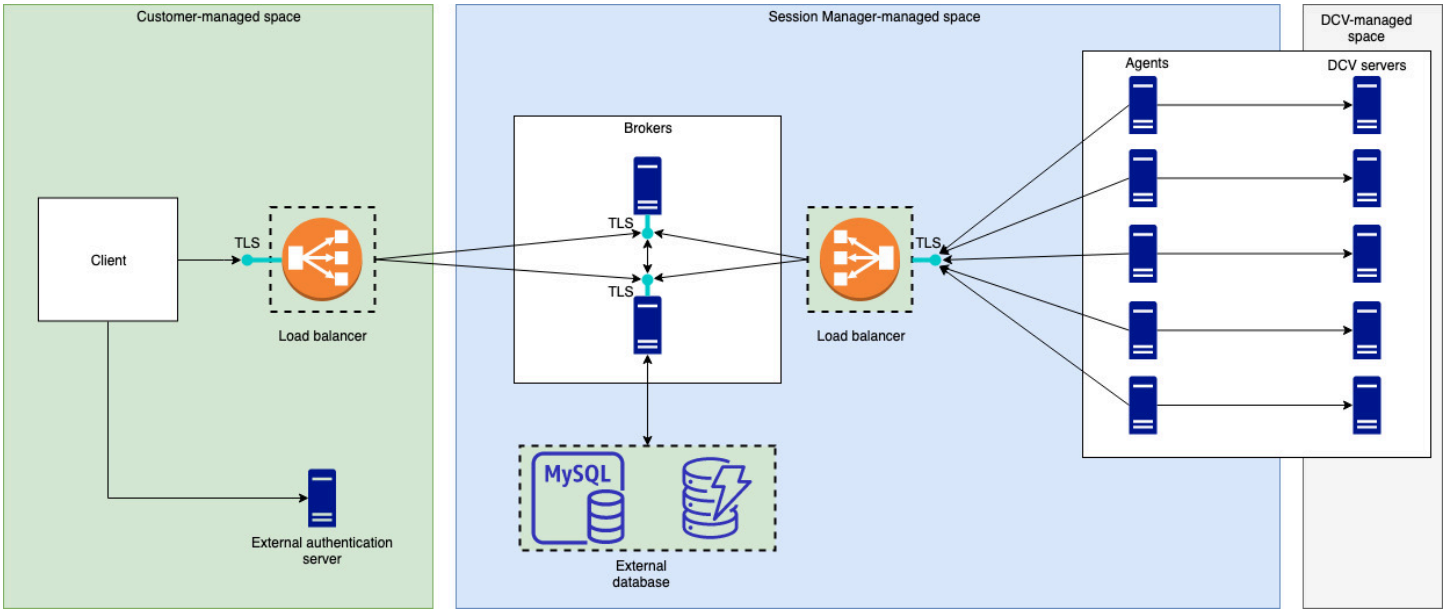
운영 체제	• Amazon Linux 2
	• CentOS 7.6 이상
	• CentOS Stream 8
	• RHEL 7.6 이상
	• RHEL 8.x
	• Rocky Linux 8.5 이상
	• Ubuntu 20.04
	• Ubuntu 22.04
		• Windows
		  • Windows Server 2019
		  • Windows Server 2016
		  • Windows Server 2012 R2
		• Linux 서버
		  • Amazon Linux 2
		  • CentOS 7.6 이상
		  • CentOS Stream 8
		  • RHEL 7.6 이상
		  • RHEL 8.x
		  • Rocky Linux 8.5 이상
		  • Ubuntu 20.04
		  • Ubuntu 22.04
		  • SUSE Linux Enterprise 12 SP4 이상
		  • SUSE Linux Enterprise 15

아키텍처	• 64비트 x86 • 64비트 ARM	• 64비트 x86 • 64비트 ARM(Amazon Linux 2, CentOS 7.x/8.x, RHEL 7.x/8.x만 해당) • 64비트 ARM(Ubuntu 22.04)

	브로커	Agent
메모리	8GB	4GB
NICE DCV 버전	NICE DCV 2020.2 이상	NICE DCV 2020.2 이상
추가 요구 사항	Java 11	-

네트워킹 및 연결 요구 사항

다음 다이어그램은 세션 관리자의 네트워킹 및 연결 요구 사항에 대한 종합적 개요를 제공합니다.



브로커는 별도의 호스트에 설치해야 하지만 NICE DCV 서버의 에이전트와 네트워크로 연결되어 있어야 합니다. 가용성을 높이기 위해 브로커를 여러 개 사용하는 경우, 각 브로커를 별도의 호스트에 설치 및 구성하고 하나 이상의 로드 밸런서를 사용하여 클라이언트와 브로커, 브로커와 에이전트 간의 트래픽을 관리해야 합니다. 또한 브로커는 NICE DCV 서버 및 세션에 대한 정보를 교환하기 위해 서로 통신할 수 있어야 합니다. 브로커는 외부 데이터베이스에 키와 상태 데이터를 저장하고 재부팅 또는 종료 후에 이 정보를 사용할 수 있습니다. 이렇게 하면 중요한 브로커 정보를 외부 데이터베이스에 보관하여 손실될 위험을 줄일 수 있습니다. 이러한 정보는 나중에 검색할 수 있습니다. 이 데이터베이스를 사용하기로 선택한 경우 외부 데이터베이스를 설정하고 브로커를 구성해야 합니다. DynamoDB, MariaDB 및 MySQL이 지원됩니다. 브로커 구성 파일에 구성 파라미터가 나열되어 있습니다.

에이전트는 브로커와 안전하고 영구적인 양방향 HTTPS 연결을 시작할 수 있어야 합니다.

API를 호출하려면 클라이언트 또는 프런트엔드 애플리케이션이 브로커에 액세스할 수 있어야 합니다. 클라이언트는 인증 서버에도 액세스할 수 있어야 합니다.

NICE DCV 세션 매니저 설정

다음 섹션에서는 단일 브로커 및 여러 에이전트와 함께 세션 관리자를 설치하는 방법을 설명합니다. 여러 브로커를 사용하여 확장성과 성능을 개선할 수 있습니다. 자세한 정보는 스케일링 세션 관리자을 참조하세요.

NICE DCV 세션 관리자를 설정하려면 다음 작업을 수행합니다.

단계

- 1단계: NICE DCV 서버 준비
- 2단계: NICE DCV 세션 관리자 브로커 설정
- 3단계: NICE DCV 세션 관리자 에이전트 설정
- 4단계: 브로커를 인증 서버로 사용하도록 NICE DCV 서버 구성
- 5단계: 설치 확인

1단계: NICE DCV 서버 준비

세션 관리자를 사용할 NICE DCV 서버가 여러 개 있어야 합니다. NICE DCV 서버 설치에 대한 자세한 내용은 NICE DCV 관리자 안내서의 NICE DCV 서버 설치를 참조하세요.

Linux NICE DCV 서버에서 세션 관리자는 `dcvsmagent`라는 로컬 서비스 사용자를 사용합니다. 이 사용자는 세션 관리자 에이전트가 설치될 때 자동으로 생성됩니다. NICE DCV가 다른 사용자를 대신하여 작업을 수행할 수 있도록 이 서비스 사용자에게 관리자 권한을 부여해야 합니다. 세션 관리자 서비스 사용자에게 관리자 권한을 부여하려면 다음과 같이 하세요.

Linux NICE DCV 서버의 로컬 서비스 사용자를 추가하려면 다음과 같이 하세요.

1. 원하는 텍스트 편집기에서 `/etc/dcv/dcv.conf` 파일을 엽니다.
2. `[security]` 섹션에 `administrators` 파라미터를 추가하고 세션 관리자 사용자를 지정합니다. 예:

   ```
   [security]
   administrators=["dcvsmagent"]
   ```

3. 파일을 저장하고 닫습니다.
4. NICE DCV 서버를 중지하고 다시 시작합니다.

세션 관리자는 NICE DCV 서버에 이미 있는 사용자를 대신하여 NICE DCV 세션을 생성할 수만 있습니다. 존재하지 않는 사용자에 대한 세션 생성을 요청하면 요청이 실패합니다. 따라서 대상이 되는 각 최종 사용자가 NICE DCV 서버에 유효한 시스템 사용자를 보유하고 있는지 확인해야 합니다.

> **Tip**
>
> 여러 브로커 호스트 또는 NICE DCV 서버를 에이전트와 함께 사용하려는 경우 다음 단계를 수행하여 구성이 완료된 호스트의 Amazon Machine Image(AMI)를 생성한 다음, AMI를 사용하여 나머지 브로커 및 NICE DCV 서버를 실행하여 브로커 하나와 NICE DCV 서버 1개만 에이전트와 함께 구성하는 것이 좋습니다. 또는 AWS Systems Manager를 사용하여 여러 인스턴스에서 원격으로 명령을 실행할 수 있습니다.

2단계: NICE DCV 세션 관리자 브로커 설정

브로커는 Linux 호스트에 설치해야 합니다. 지원되는 Linux 배포에 대한 자세한 내용은 NICE DCV 세션 관리자 요구 사항 섹션을 참조하세요. 에이전트 및 NICE DCV 서버 호스트와 분리된 호스트에 브로커를 설치합니다. 호스트는 다른 프라이빗 네트워크에 설치할 수 있지만 에이전트에 연결하여 에이전트와 통신할 수 있어야 합니다.

브로커를 설치 및 시작하려면 다음과 같이 하세요.

1. 브로커를 설치하려는 호스트에 연결합니다.
2. 패키지는 안전한 GPG 서명으로 디지털 서명됩니다. 패키지 관리자가 패키지 서명을 확인할 수 있도록 하려면 NICE GPG 키를 가져와야 합니다. 다음 명령을 실행하여 NICE GPG 키를 가져옵니다.

   - Amazon Linux 2, RHEL, CentOS, Rocky Linux

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

   - Ubuntu

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

3. 설치 패키지를 다운로드합니다.

   - Amazon Linux 2, RHEL 7.x, CentOS 7.x

```
$ wget https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerBrokers/nice-
dcv-session-manager-broker-2023.1.410-1.el7.noarch.rpm
```

- RHEL 8.x, CentOS Stream 8, Rocky Linux 8.x

```
$ wget https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerBrokers/nice-
dcv-session-manager-broker-2023.1.410-1.el8.noarch.rpm
```

- Ubuntu 20.04

```
$ wget https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerBrokers/nice-
dcv-session-manager-broker_2023.1.410-1_all.ubuntu2004.deb
```

- Ubuntu 22.04

```
$ wget https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerBrokers/nice-
dcv-session-manager-broker_2023.1.410-1_all.ubuntu2204.deb
```

4. 패키지를 설치합니다.

- Amazon Linux 2, RHEL 7.x, CentOS 7.x

```
$ sudo yum install -y ./nice-dcv-session-manager-
broker-2023.1.410-1.el7.noarch.rpm
```

- RHEL 8.x, Stream CentOS 8, Rocky Linux 8.x

```
$ sudo yum install -y ./nice-dcv-session-manager-
broker-2023.1.410-1.el8.noarch.rpm
```

- Ubuntu 20.04

```
$ sudo apt install -y ./nice-dcv-session-manager-
broker_2023.1.410-1_all.ubuntu2004.deb
```

- Ubuntu 22.04

```
$ sudo apt install -y ./nice-dcv-session-manager-
broker_2023.1.410-1_all.ubuntu2204.deb
```

5. 기본 Java 환경 버전이 11인지 확인하세요.

```
$ java -version
```

그렇지 않은 경우 브로커가 올바른 Java 버전을 대상으로 지정하는 데 사용할 Java 홈 디렉터리를 명시적으로 설정할 수 있습니다. 이렇게 하면 브로커 구성 파일에서 `broker-java-home` 파라미터를 설정하는 작업이 완료됩니다. 자세한 정보는 브로커 구성 파일을 참조하세요.

6. 브로커 서비스를 시작하고 인스턴스가 시작될 때마다 서비스가 자동으로 시작되는지 확인합니다.

```
$  sudo systemctl start dcv-session-manager-broker && sudo systemctl enable dcv-
session-manager-broker
```

7. 브로커의 자체 서명 인증서 사본을 사용자 디렉터리에 배치합니다. 다음 단계에서 에이전트를 설치할 때 필요합니다.

```
sudo cp /var/lib/dcvsmbroker/security/dcvsmbroker_ca.pem $HOME
```

3단계: NICE DCV 세션 관리자 에이전트 설정

에이전트는 플릿의 모든 NICE DCV 서버 호스트에 설치되어야 합니다. 에이전트는 Windows 및 Linux 서버 모두에 설치할 수 있습니다. 지원되는 운영 체제에 대한 자세한 내용은 NICE DCV 세션 관리자 요구 사항 섹션을 참조하세요.

필수 조건

에이전트를 설치하기 전에 NICE DCV 서버를 호스트에 설치해야 합니다.

Linux host

> **Note**
>
> 세션 관리자 에이전트는 요구 사항에 나열된 Linux 배포판 및 아키텍처에서 사용할 수 있습니다.
>
> 다음은 64비트 x86 호스트에 에이전트를 설치하기 위한 안내입니다. *64## ARM #### ## ### ##### x86_64# # ######.* aarch64 *#### ## amd64# # ######.* arm64

Linux 호스트에 에이전트를 설치하려면 다음과 같이 하세요.

1. 패키지는 안전한 GPG 서명으로 디지털 서명됩니다. 패키지 관리자가 패키지 서명을 확인할 수 있도록 하려면 NICE GPG 키를 가져와야 합니다. 다음 명령을 실행하여 NICE GPG 키를 가져옵니다.

   - Amazon Linux 2, RHEL, CentOS, 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. 설치 패키지를 다운로드합니다.

   - Amazon Linux 2, RHEL 7.x, CentOS 7.x

     ```
     $  wget https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerAgents/
     nice-dcv-session-manager-agent-2023.1.732-1.el7.x86_64.rpm
     ```

   - RHEL 8.x, CentOS Stream 8, Rocky Linux 8.x

     ```
     $  wget https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerAgents/
     nice-dcv-session-manager-agent-2023.1.732-1.el8.x86_64.rpm
     ```

   - Ubuntu 20.04

     ```
     $  wget https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerAgents/
     nice-dcv-session-manager-agent_2023.1.732-1_amd64.ubuntu2004.deb
     ```

   - Ubuntu 22.04

     ```
     $  wget https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerAgents/
     nice-dcv-session-manager-agent_2023.1.732-1_amd64.ubuntu2204.deb
     ```

   - SUSE Linux Enterprise 12

```
$ curl -O https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerAgents/
nice-dcv-session-manager-agent-2023.1.732-1.sles12.x86_64.rpm
```

- SUSE Linux Enterprise 15

```
$ curl -O https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerAgents/
nice-dcv-session-manager-agent-2023.1.732-1.sles15.x86_64.rpm
```

3. 패키지를 설치합니다.

- Amazon Linux 2, RHEL 7.x, CentOS 7.x

```
$  sudo yum install -y ./nice-dcv-session-manager-
agent-2023.1.732-1.el7.x86_64.rpm
```

- RHEL 8.x, CentOS Stream 8, Rocky Linux 8.x

```
$  sudo yum install -y ./nice-dcv-session-manager-
agent-2023.1.732-1.el8.x86_64.rpm
```

- Ubuntu 20.04

```
$  sudo apt install ./nice-dcv-session-manager-
agent_2023.1.732-1_amd64.ubuntu2004.deb
```

- Ubuntu 22.04

```
$  sudo apt install ./nice-dcv-session-manager-
agent_2023.1.732-1_amd64.ubuntu2204.deb
```

- SUSE Linux Enterprise 12

```
$ sudo zypper install ./nice-dcv-session-manager-
agent-2023.1.732-1.sles12.x86_64.rpm
```

- SUSE Linux Enterprise 15

```
$ sudo zypper install ./nice-dcv-session-manager-
agent-2023.1.732-1.sles15.x86_64.rpm
```

4. 브로커의 자체 서명 인증서 사본(이전 단계에서 복사한 사본)을 에이전트의 `/etc/dcv-session-manager-agent/` 디렉터리에 저장합니다.
5. 선호하는 텍스트 편집기를 사용하여 `/etc/dcv-session-manager-agent/agent.conf`를 열고 다음과 같이 진행합니다.

   - `broker_host`의 경우 브로커가 설치된 호스트의 DNS 이름을 지정합니다.

     > **Important**
     >
     > 브로커가 Amazon EC2 인스턴스에서 실행 중인 경우 `broker_host`에 인스턴스의 프라이빗 IPv4 주소를 지정해야 합니다.

   - (선택 사항) `broker_port`에 브로커와 통신하는 데 사용할 포트를 지정합니다. 기본적으로 에이전트와 브로커는 8445 포트를 통해 통신합니다. 다른 포트를 사용해야 하는 경우에만 이 값을 변경합니다. 변경할 경우 브로커가 동일한 포트를 사용하도록 구성되어 있는지 확인하세요.
   - `ca_file`에는 이전 단계에서 복사한 인증서 파일의 전체 경로를 지정합니다. 예:

     ```
     ca_file = '/etc/dcv-session-manager-agent/broker_cert.pem'
     ```

     또는 TLS 확인을 비활성화하려면 `tls_strict`를 `false`로 설정합니다.
6. 파일을 저장하고 닫습니다.
7. 다음 명령을 실행하여 에이전트를 시작합니다.

   ```
   $ sudo systemctl start dcv-session-manager-agent
   ```

Windows host

Windows 호스트에 에이전트를 설치하려면 다음과 같이 하세요.

1. 에이전트 설치 프로그램을 다운로드합니다.
2. 설치 관리자를 실행합니다. 시작 화면에서 다음을 선택합니다.
3. EULA 화면에서 라이선스 계약을 주의 깊게 읽어보고, 약관에 동의하는 경우 약관에 동의함을 선택하고 다음을 클릭합니다.
4. 설치를 시작하려면 설치를 클릭합니다.

5. 브로커의 자체 서명 인증서 사본(이전 단계에서 복사한 사본)을 에이전트의 `C:\Program Files\NICE\DCVSessionManagerAgent\conf\` 폴더에 저장합니다.
6. 선호하는 텍스트 편집기를 사용하여 `C:\Program Files\NICE\DCVSessionManagerAgent\conf\agent.conf`를 열고 다음과 같이 진행합니다.

   - `broker_host`의 경우 브로커가 설치된 호스트의 DNS 이름을 지정합니다.

     > **Important**
     >
     > 브로커가 Amazon EC2 인스턴스에서 실행 중인 경우 `broker_host`에 인스턴스의 프라이빗 IPv4 주소를 지정해야 합니다.

   - (선택 사항) `broker_port`에 브로커와 통신하는 데 사용할 포트를 지정합니다. 기본적으로 에이전트와 브로커는 8445 포트를 통해 통신합니다. 다른 포트를 사용해야 하는 경우에만 이 값을 변경합니다. 변경할 경우 브로커가 동일한 포트를 사용하도록 구성되어 있는지 확인하세요.
   - `ca_file`에는 이전 단계에서 복사한 인증서 파일의 전체 경로를 지정합니다. 예:

     ```
     ca_file = 'C:\Program Files\NICE\DCVSessionManagerAgent\conf\broker_cert.pem'
     ```

     또는 TLS 확인을 비활성화하려면 `tls_strict`를 `false`로 설정합니다.
7. 파일을 저장하고 닫습니다.
8. 변경 사항이 적용되도록 에이전트 서비스를 중지했다가 다시 시작합니다. 명령 프롬프트에서 다음 명령을 실행합니다.

   ```
   C:\> sc stop DcvSessionManagerAgentService
   ```

   ```
   C:\> sc start DcvSessionManagerAgentService
   ```

4단계: 브로커를 인증 서버로 사용하도록 NICE DCV 서버 구성

브로커를 클라이언트 연결 토큰의 유효성을 검사하는 외부 인증 서버로 사용하도록 NICE DCV 서버를 구성합니다. 또한 브로커의 자체 서명 CA를 신뢰하도록 NICE DCV 서버를 구성해야 합니다.

Linux NICE DCV server

Linux NICE DCV 서버의 로컬 서비스 사용자를 추가하려면 다음과 같이 하세요.

1. 원하는 텍스트 편집기에서 `/etc/dcv/dcv.conf` 파일을 엽니다.
2. `ca-file` 및 `auth-token-verifier` 파라미터를 `[security]` 섹션에 추가합니다.

   `ca-file`에 이전 단계에서 호스트에 복사한 브로커의 자체 서명 CA 경로를 지정합니다.

   `auth-token-verifier`에 브로커의 토큰 식별자 URL을 `https://`*`broker_ip_or_dns`*`:`*`port`*`/agent/validate-authentication-token` 형식으로 지정합니다. 브로커-에이전트 통신에 사용되는 포트를 지정합니다. 기본값은 8445입니다. Amazon EC2 인스턴스에서 브로커를 실행하는 경우 프라이빗 DNS 또는 프라이빗 IP 주소를 사용해야 합니다.

   예

   ```
   [security]
   ca-file="/etc/dcv-session-manager-agent/broker_cert.pem"
   auth-token-verifier="https://my-sm-broker.com:8445/agent/validate-
   authentication-token"
   ```

3. 파일을 저장하고 닫습니다.
4. NICE DCV 서버를 중지하고 다시 시작합니다. 자세한 내용은 NICE DCV 관리자 안내서의 NICE DCV 서버 중지 및 NICE DCV 서버 시작을 참조하세요.

Windows NICE DCV server

Windows NICE DCV 서버

1. Windows 레지스트리 편집기를 열고 HKEY_USERS/S-1-5-18/Software/GSettings/com/nicesoftware/dcv/security/ 키로 이동합니다.
2. ca-file 파라미터를 엽니다. 값 데이터의 경우 이전 단계에서 호스트에 복사한 브로커의 자체 서명 CA 경로를 지정하세요.

   > **Note**
   >
   > 파라미터가 없는 경우 새 문자열 파라미터를 만들고 이름을 `ca-file`로 지정합니다.

3. 파라미터를 엽니다. auth-token-verifier 값 데이터의 경우 브로커의 토큰 식별자 URL을 `https://broker_ip_or_dns:port/agent/validate-authentication-token` 형식으로 지정합니다. 브로커-에이전트 통신에 사용되는 포트를 지정합니다. 기본값은 8445입니다. Amazon EC2 인스턴스에서 브로커를 실행하는 경우 프라이빗 DNS 또는 프라이빗 IP 주소를 사용해야 합니다.

   > **Note**
   >
   > 파라미터가 없는 경우 새 문자열 파라미터를 만들고 이름을 `auth-token-verifier`로 지정합니다.

4. 확인을 선택하고 Windows 레지스트리 편집기를 닫습니다.
5. NICE DCV 서버를 중지하고 다시 시작합니다. 자세한 내용은 NICE DCV 관리자 안내서의 NICE DCV 서버 중지 및 NICE DCV 서버 시작을 참조하세요.

5단계: 설치 확인

주제

에이전트 확인

브로커와 에이전트를 설치한 후, 에이전트가 실행 중이고 브로커에 연결할 수 있는지 확인하세요.

Linux 에이전트 호스트

실행할 명령은 버전에 따라 다릅니다.

- 버전 2022.0 이상

  에이전트 호스트에서 다음 명령을 실행합니다.

  ```
  $ grep 'sessionsUpdateResponse' /var/log/dcv-session-manager-agent/agent.log | tail
   -1 | grep -o success
  ```

- 버전 2022.0 이하

에이전트 호스트에서 다음 명령을 실행하고 현재 연도, 월, 일을 지정합니다.

```
$ grep 'sessionsUpdateResponse' /var/log/dcv-session-manager-agent/
agent.log.yyyy-mm-dd | tail -1 | grep -o success
```

예

```
$ grep 'sessionsUpdateResponse' /var/log/dcv-session-manager-agent/
agent.log.2020-11-19 | tail -1 | grep -o success
```

에이전트가 실행 중이고 브로커에 연결할 수 있는 경우 명령이 success 값을 반환해야 합니다.

명령이 다른 출력을 반환하는 경우 에이전트 로그 파일에서 자세한 내용을 확인하세요. 로그 파일은 /var/log/dcv-session-manager-agent/ 위치에 있습니다.

Windows 에이전트 호스트

C:\ProgramData\NICE\DCVSessionManagerAgent\log에 있는 에이전트 로그 파일을 엽니다.

로그 파일에 아래와 비슷한 행이 있는 경우 에이전트가 실행 중인 것이므로 브로커에 연결할 수 있습니다.

```
2020-11-02 12:38:03,996919  INFO ThreadId(05) dcvsessionmanageragent::agent:Processing
 broker message "{\n  \"sessionsUpdateResponse\" : {\n    \"requestId\" :
 \"69c24a3f5f6d4f6f83ffbb9f7dc6a3f4\",\n    \"result\" : {\n      \"success\" : true\n
   }\n  }\n}"
```

로그 파일에 비슷한 행이 없는 경우 로그 파일에 오류가 있는지 검사해 보세요.

브로커 확인

브로커와 에이전트를 설치한 후에는 브로커가 실행 중이고 사용자 및 프런트엔드 애플리케이션에서 연결할 수 있는지 확인하세요.

브로커에 연결할 수 있는 컴퓨터에서 다음 명령을 실행합니다.

```
$ curl -X GET https://broker_host_ip:port/sessionConnectionData/aSession/aOwner --
insecure
```

확인이 성공하면 브로커는 다음을 반환합니다.

```
{
    "error": "No authorization header"
}
```

NICE DCV 세션 관리자 구성

이 섹션에서는 세션 관리자의 고급 구성을 수행하는 방법을 설명합니다.

주제

스케일링 세션 관리자

고가용성을 활성화하고 성능을 향상시키기 위해 여러 에이전트 및 브로커를 사용하도록 세션 관리자를 구성할 수 있습니다. 여러 에이전트와 브로커를 사용하려는 경우 에이전트 및 브로커 호스트를 하나만 설치 및 구성하고, 해당 호스트에서 Amazon Machines Image(AMI)를 생성한 다음 AMI에서 나머지 호스트를 시작하는 것이 좋습니다.

기본적으로 세션 관리자는 추가 구성 없이 여러 에이전트를 사용할 수 있도록 지원합니다. 그러나 여러 브로커를 사용하려는 경우에는 로드 밸런서를 사용하여 프런트엔드 클라이언트와 브로커 간, 브로커와 에이전트 간의 트래픽 균형을 조정해야 합니다. 로드 밸런서 설정 및 구성은 전적으로 사용자가 소유하고 관리합니다.

다음 섹션에서는 Application Load Balancer와 함께 여러 호스트를 사용하도록 세션 관리자를 구성하는 방법을 설명합니다.

단계

1단계: 인스턴스 프로파일 생성

Elastic Load Balancing API를 사용할 권한을 부여하는 브로커 및 에이전트 호스트에 인스턴스 프로파일을 연결해야 합니다. 자세한 내용은 Amazon EC2 사용 설명서의 Amazon EC2의 IAM 역할을 참조하세요.

인스턴스 프로파일 생성

1. 인스턴스 프로필에서 사용할 권한을 정의하는 AWS Identity and Access Management (IAM) 역할을 생성합니다. 다음 신뢰 정책을 사용합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

그리고 다음 정책을 연결합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ec2:DescribeInstances"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Action": [
        "elasticloadbalancing:DescribeTargetHealth"
      ],
      "Effect": "Allow",
      "Resource": "*"
```

```
    }
  ]
}
```

자세한 내용은 IAM 사용 설명서의 IAM 역할 생성을 참조하세요.

2. 새 인스턴스 프로파일을 생성합니다. 자세한 내용은 AWS CLI 명령 참조에서 create-instance-profile을 참조하세요.
3. 인스턴스 프로파일에 IAM 역할을 추가합니다. 자세한 내용은 AWS CLI 명령 참조에서 add-role-to-instance-profile을 참조하세요.
4. 인스턴스 프로파일을 브로커 호스트에 연결합니다. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스에 IAM 역할 연결을 참조하십시오.

2단계: 로드 밸런서용 SSL 인증서 준비

로드 밸런서 리스너에서 HTTPS를 사용할 때 로드 밸런서에 SSL 인증서를 반드시 배포해야 합니다. 로드 밸런서는 이 인증서를 사용해 연결을 종료하고 대상으로 전송하기 전에 클라이언트의 요청을 해독합니다.

SSL 인증서를 준비하려면 다음과 같이 하세요.

1. 사설 인증 기관 (CA) AWS 인증서 관리자 사설 인증 기관 (ACM PCA) 을 생성합니다. 자세한 내용은 AWS Certificate Manager 사설 인증 기관 사용 설명서의 CA 생성 절차를 참조하십시오.
2. CA를 설치합니다. 자세한 내용은 Certificate Manager 사설 인증 기관 사용 AWS 설명서의 루트 CA 인증서 설치를 참조하십시오.
3. CA에서 서명한 새 프라이빗 인증서를 요청합니다. 도메인 이름에는 `*.region.elb.amazonaws.com`을 사용하고 로드 밸런서를 생성하려는 리전을 지정합니다. 자세한 내용은 Certificate Manager 사설 인증 기관 사용 설명서의 사설 인증서 요청을 참조하십시오. AWS

3단계: 브로커 Application Load Balancer 생성

Application Load Balancer를 생성하여 프런트엔드 클라이언트와 브로커 간의 트래픽 균형을 조정합니다.

로드 밸런서를 생성하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.

탐색 창에서 로드 밸런서를 선택한 다음 로드 밸런서 생성을 선택합니다. 로드 밸런서 유형으로 Application Load Balancer를 선택합니다.

2. 1단계: 로드 밸런서 구성에서 다음을 수행합니다.

   a. 이름에 로드 밸런서를 설명하는 이름을 입력합니다.

   b. 스키마에서 인터넷 경계를 선택합니다.

   c. 로드 밸런서 프로토콜에 HTTPS를 선택하고 로드 밸런서 포트에 8443을 입력합니다.

   d. VPC의 경우 사용할 VPC를 선택한 다음 해당 VPC에 있는 모든 서브넷을 선택합니다.

   e. 다음을 선택합니다.

3. 2단계: 보안 설정 구성에서 다음을 수행합니다.

   a. 인증서 유형에서 ACM에서 인증서 선택을 선택합니다.

   b. 인증서 이름에는 이전에 요청한 프라이빗 인증서를 선택합니다.

   c. 다음을 선택합니다.

4. 3단계: 보안 그룹 구성의 경우 새 보안 그룹을 생성하거나 HTTPS 및 포트 8443을 통해 프런트엔드 클라이언트와 브로커 간의 인바운드 및 아웃바운드 트래픽을 허용하는 기존 보안 그룹을 선택합니다.

   다음을 선택합니다.

5. 4단계: 라우팅 구성에서 다음을 수행합니다.

   a. 대상 그룹의 경우 새 대상 그룹을 선택합니다.

   b. 이름에 대상 그룹의 이름을 입력합니다.

   c. 대상 유형에서 인스턴스를 선택합니다.

   d. 프로토콜에서 HTTPS를 선택합니다. 포트에서 8443를 입력합니다. 프로토콜 버전에서 HTTP1을 선택합니다.

   e. 상태 확인 프로토콜의 경우 HTTPS를 선택하고 경로에 `/health`를 입력합니다.

   f. 다음을 선택합니다.

6. 5단계: 대상 등록에서 다음을 선택합니다.

7. 생성을 선택합니다.

4단계: 브로커 시작

초기 브로커를 생성하고, 로드 밸런서를 사용하도록 구성하고, 브로커에서 AMI를 생성한 다음 AMI를 사용하여 나머지 브로커를 시작합니다. 이렇게 하면 모든 브로커가 동일한 CA와 동일한 로드 밸런서 구성을 사용하도록 구성할 수 있습니다.

브로커를 시작하려면 다음과 같이 하세요.

1. 초기 브로커 호스트를 시작하고 구성합니다. 브로커 설치 및 구성에 대한 자세한 내용은 2단계: NICE DCV 세션 관리자 브로커 설정 섹션을 참조하세요.

   > **Note**
   >
   > Application Load Balancer를 사용하기 때문에 브로커의 자체 서명 인증서는 필요하지 않습니다.

2. 브로커에 연결하고 선호하는 텍스트 편집기를 사용하여 `/etc/dcv-session-manager-broker/session-manager-broker.properties`를 열고 다음과 같이 진행합니다.

   a. 행 시작 부분에 해시(#)를 넣어 `broker-to-broker-discovery-addresses` 파라미터를 주석 처리합니다.

   b. `broker-to-broker-discovery-aws-region`에 Application Load Balancer를 생성한 리전을 입력합니다.

   c. `broker-to-broker-discovery-aws-alb-target-group-arn`에 브로커 로드 밸런서와 연결된 대상 그룹의 ARN을 입력합니다.

   d. 파일을 저장하고 닫습니다.

3. 브로커 인스턴스를 중지합니다.
4. 중지된 Broker 인스턴스에서 AMI를 생성합니다. 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 인스턴스에서 Linux AMI 생성을 참조하세요.
5. AMI를 사용하여 나머지 브로커를 시작합니다.
6. 생성한 인스턴스 프로파일을 모든 브로커 인스턴스에 할당합니다.
7. 브로커 간 네트워크 트래픽과 브로커 및 로드 밸런서 간 네트워크 트래픽을 모든 브로커 인스턴스에 허용하는 보안 그룹을 할당합니다. 네트워크 포트에 대한 자세한 내용은 브로커 구성 파일을 참조하세요.
8. 모든 브로커 인스턴스를 브로커 로드 밸런서의 대상으로 등록합니다. 자세한 내용은 Application Load Balancer 사용 설명서에서 대상 그룹에 대상 등록을 참조하세요.

5단계: 에이전트 Application Load Balancer 생성

Application Load Balancer를 생성하여 에이전트와 브로커의 균형을 조정합니다.

로드 밸런서를 생성하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.

   탐색 창에서 로드 밸런서를 선택한 다음 로드 밸런서 생성을 선택합니다. 로드 밸런서 유형으로 Application Load Balancer를 선택합니다.
2. 1단계: 로드 밸런서 구성에서 다음을 수행합니다.

   a. 이름에 로드 밸런서를 설명하는 이름을 입력합니다.

   b. 스키마에서 인터넷 경계를 선택합니다.

   c. 로드 밸런서 프로토콜에 HTTPS를 선택하고 로드 밸런서 포트에 8445을 입력합니다.

   d. VPC의 경우 사용할 VPC를 선택한 다음 해당 VPC에 있는 모든 서브넷을 선택합니다.

   e. 다음을 선택합니다.
3. 2단계: 보안 설정 구성에서 다음을 수행합니다.

   a. 인증서 유형에서 ACM에서 인증서 선택을 선택합니다.

   b. 인증서 이름에는 이전에 요청한 프라이빗 인증서를 선택합니다.

   c. 다음을 선택합니다.
4. 3단계: 보안 그룹 구성의 경우 새 보안 그룹을 생성하거나 HTTPS 및 포트 8445를 통해 에이전트와 브로커 간의 인바운드 및 아웃바운드 트래픽을 허용하는 기존 보안 그룹을 선택합니다.

   다음을 선택합니다.
5. 4단계: 라우팅 구성에서 다음을 수행합니다.

   a. 대상 그룹에 대해 새 목표 그룹을 선택합니다.

   b. 이름에 대상 그룹의 이름을 입력합니다.

   c. 대상 유형에서 인스턴스를 선택합니다.

   d. 프로토콜에서 HTTPS를 선택합니다. 포트에서 8445를 입력합니다. 프로토콜 버전에서 HTTP1을 선택합니다.

   e. 상태 확인 프로토콜의 경우 HTTPS를 선택하고 경로에 `/health`를 입력합니다.

   f. 다음을 선택합니다.

6. 5단계: 대상 등록의 경우 모든 브로커 인스턴스를 선택하고 등록된 항목에 추가를 선택합니다. 다음: 검토를 선택합니다.
7. 생성을 선택합니다.

6단계: 에이전트 시작

초기 에이전트를 생성하고, 로드 밸런서를 사용하도록 구성하고, 에이전트에서 AMI를 생성한 다음 AMI를 사용하여 나머지 에이전트를 시작합니다. 이렇게 하면 모든 에이전트가 동일한 로드 밸런서 구성을 사용하도록 구성할 수 있습니다.

에이전트를 시작하려면 다음과 같이 하세요.

1. NICE DCV 서버를 준비합니다. 자세한 정보는 1단계: NICE DCV 서버 준비을 참조하세요.
2. 2단계: 로드 밸런서용 SSL 인증서 준비에서 생성한 CA 퍼블릭 키의 사본을 배치합니다. 모든 사용자가 읽을 수 있는 디렉터리를 선택하거나 생성합니다. CA 공개 키 파일은 모든 사용자가 읽을 수 있어야 합니다.
3. 에이전트를 설치 및 구성합니다. 설치 및 구성에 대한 자세한 내용은 3단계: NICE DCV 세션 관리자 에이전트 설정 섹션을 참조하세요.

   > **Important**
   >
   > 에이전트 구성 파일 수정 시:
   >
   > - `broker_host` 파라미터에는 에이전트 로드 밸런서의 DNS를 입력합니다.
   > - `ca_file` 파라미터에는 이전 단계에서 생성한 CA 퍼블릭 키 파일의 경로를 입력합니다.

4. 브로커를 인증 서버로 사용하도록 NICE DCV 서버를 구성합니다. 자세한 정보는 4단계: 브로커를 인증 서버로 사용하도록 NICE DCV 서버 구성을 참조하세요.

   > **Important**
   >
   > NICE DCV 서버 구성 파일 수정 시:
   >
   > - `ca-file` 파라미터에는 이전 단계에서 사용한 CA 퍼블릭 키 파일의 동일한 경로를 입력합니다.

- `auth-token-verifier` 파라미터에는 *`broker_ip_or_dns`*에 대한 에이전트 로드 밸런서의 DNS를 사용합니다.

5. 에이전트 인스턴스를 중지합니다.
6. 중지된 에이전트 인스턴스에서 AMI를 생성합니다. 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 인스턴스에서 Linux AMI 생성을 참조하세요.
7. AMI를 사용하여 나머지 에이전트를 시작하고 생성한 인스턴스 프로파일을 모든 에이전트에 할당합니다.
8. 에이전트와 로드 밸런서 간 네트워크 트래픽을 모든 에이전트 인스턴스에 허용하는 보안 그룹을 할당합니다. 네트워크 포트에 대한 자세한 내용은 에이전트 구성 파일을 참조하세요.

태그를 사용하여 NICE DCV 서버를 대상으로 지정

세션 관리자 에이전트에 사용자 지정 태그를 할당하면 에이전트와 연결된 NICE DCV 서버를 식별하고 분류하는 데 도움이 될 수 있습니다. 새 NICE DCV 세션을 생성할 때 해당 에이전트에 할당된 태그를 기반으로 NICE DCV 서버 그룹을 대상으로 지정할 수 있습니다. 에이전트 태그를 기반으로 NICE DCV 서버를 대상으로 지정하는 방법에 대한 자세한 내용은 세션 관리자 개발자 안내서의 CreateSessionRequests 항목을 참조하세요.

태그는 태그 키와 값 쌍으로 구성되며 사용 사례 또는 환경에 적합한 모든 정보 쌍을 사용할 수 있습니다. 호스트의 하드웨어 구성에 따라 에이전트에 태그를 지정하도록 선택할 수 있습니다. 예를 들어, `ram=4GB` 값의 4GB 메모리가 있는 호스트로 모든 에이전트에 태그를 지정할 수 있습니다. 또는 용도에 따라 에이전트에 태그를 지정할 수도 있습니다. 예를 들어, `purpose=production` 값의 프로덕션 호스트에서 실행되는 모든 에이전트에 태그를 지정할 수 있습니다.

에이전트에 태그를 할당하려면

1. 선호하는 텍스트 편집기를 사용하여 새 파일을 생성하고 설명이 포함된 이름(예:`agent_tags.toml`)을 지정합니다. 파일 유형은 `.toml`이어야 하고, 파일 내용은 TOML 파일 형식으로 지정해야 합니다.
2. 파일에서 `key=value` 형식을 사용하여 새 행에 새 태그 키와 값 쌍을 각각 추가합니다. 예:

```
tag1="abc"
tag2="xyz"
```

3. 에이전트 구성 파일(Linux용 `/etc/dcv-session-manager-agent/agent.conf` 또는 Windows용 `C:\Program Files\NICE\DCVSessionManagerAgent\conf\agent.conf`)을 엽니다. `tags_folder`에 태그 파일이 있는 디렉터리의 경로를 지정합니다.

   디렉터리에 여러 태그 파일이 있는 경우, 파일 전체에 정의된 모든 태그가 에이전트를 적용합니다. 파일은 알파벳순으로 읽힙니다. 키가 같은 태그가 여러 파일에 포함된 경우, 마지막으로 읽은 파일의 값으로 값을 덮어씁니다.

4. 파일을 저장하고 닫습니다.
5. 에이전트를 중지한 후 다시 시작합니다.

   - Windows

   ```
   C:\> sc stop DcvSessionManagerAgentService
   ```

   ```
   C:\> sc start DcvSessionManagerAgentService
   ```

   - Linux

   ```
   $ sudo systemctl stop dcv-session-manager-agent
   ```

   ```
   $ sudo systemctl start dcv-session-manager-agent
   ```

외부 권한 부여 서버 구성

권한 부여 서버는 클라이언트 SDK 및 에이전트의 인증 및 권한 부여를 담당하는 서버입니다.

기본적으로 세션 관리자는 브로커를 권한 부여 서버로 사용하여 클라이언트 SDK용 OAuth 2.0 액세스 토큰과 에이전트용 소프트웨어 명령문을 생성합니다. 브로커를 권한 부여 서버로 사용하는 경우 추가 구성이 필요하지 않습니다.

브로커 대신 Amazon Cognito를 외부 권한 부여 서버로 사용하도록 세션 관리자를 구성할 수 있습니다. Amazon Cognito에 대한 자세한 내용은 Amazon Cognito 개발자 안내서를 참조하세요.

Amazon Cognito를 권한 부여 서버로 사용하려면

1. 새 Amazon Cognito 사용자 풀을 생성합니다. 사용자 풀에 대한 자세한 내용은 Amazon Cognito 개발자 안내서의 Amazon Cognito의 기능을 참조하세요.

create-user-pool 명령을 사용하고 풀 이름과 이를 생성할 리전을 지정합니다.

이 예제에서는 풀의 이름을 dcv-session-manager-client-app으로 지정하고 us-east-1에 풀을 생성합니다.

```
$ aws cognito-idp create-user-pool --pool-name dcv-session-manager-client-app --
region us-east-1
```

출력 예

```
{
    "UserPoolClient": {
        "UserPoolId": "us-east-1_QLEXAMPLE",
        "ClientName": "dcv-session-manager-client-app",
        "ClientId": "15hhd8jij74hf32f24uEXAMPLE",
        "LastModifiedDate": 1602510048.054,
        "CreationDate": 1602510048.054,
        "RefreshTokenValidity": 30,
        "AllowedOAuthFlowsUserPoolClient": false
    }
}
```

다음 단계에서 필요하므로 userPoolId 값을 기록해 둡니다.

2. 사용자 풀의 새 도메인을 생성합니다. create-user-pool-domain 명령을 사용하여 이전 단계에서 만든 사용자 풀의 도메인 이름과 userPoolId를 지정합니다.

   이 예제에서는 도메인 이름을 mydomain-544fa30f-c0e5-4a02-8d2a-a3761EXAMPLE로 지정하고 us-east-1에 도메인을 생성합니다.

   ```
   $ aws cognito-idp create-user-pool-domain --domain mydomain-544fa30f-
   c0e5-4a02-8d2a-a3761EXAMPLE --user-pool-id us-east-1_QLEXAMPLE --region us-east-1
   ```

   출력 예

   ```
   {
       "DomainDescription": {
           "UserPoolId": "us-east-1_QLEXAMPLE",
           "AWSAccountId": "123456789012",
           "Domain": "mydomain-544fa30f-c0e5-4a02-8d2a-a3761EXAMPLE",
   ```

```
        "S3Bucket": "aws-cognito-prod-pdx-assets",
        "CloudFrontDistribution": "dpp0gtexample.cloudfront.net",
        "Version": "20201012133715",
        "Status": "ACTIVE",
        "CustomDomainConfig": {}
    }
}
```

사용자 풀 도메인의 형식은 다음과 같습니다.
`https://domain_name.auth.region.amazoncognito.com` 이 예제에서 사용자 풀 도메인은 `https://mydomain-544fa30f-c0e5-4a02-8d2a-a3761EXAMPLE.auth.us-east-1.amazoncognito.com`입니다.

3. 사용자 풀 클라이언트를 생성합니다. create-user-pool-client 명령을 사용하여 생성한 사용자 풀의 `userPoolId`, 클라이언트의 이름, 사용자 풀을 생성할 리전을 지정합니다. 또한 생성 중인 사용자 풀 클라이언트에 대해 암호를 만들지 여부를 지정하는 `--generate-secret` 옵션을 포함합니다.

   이 경우 클라이언트 이름은 `dcv-session-manager-client-app`이며 `us-east-1` 리전에 생성합니다.

   ```
   $ aws cognito-idp create-user-pool-client --user-pool-id us-east-1_QLEXAMPLE --
   client-name dcv-session-manager-client-app --generate-secret --region us-east-1
   ```

   출력 예

   ```
   {
       "UserPoolClient": {
           "UserPoolId": "us-east-1_QLEXAMPLE",
           "ClientName": "dcv-session-manager-client-app",
           "ClientId": "2l9273hp6k2ut5cugg9EXAMPLE",
           "ClientSecret": "1vp5e8nec7cbf4m9me55mbmht91u61hlh0a78rq1qki1lEXAMPLE",
           "LastModifiedDate": 1602510291.498,
           "CreationDate": 1602510291.498,
           "RefreshTokenValidity": 30,
           "AllowedOAuthFlowsUserPoolClient": false
       }
   }
   ```

> **Note**
>
> `ClientId` 및 `ClientSecret` 값을 적어둡니다. 개발자가 API 요청에 대한 액세스 토큰을 요청할 때 사용할 수 있도록 이 정보를 제공해야 합니다.

4. 사용자 풀에 사용할 새로운 OAuth2.0 리소스 서버를 생성합니다. 리소스 서버는 액세스 보호가 되는 리소스의 서버로서, 액세스 토큰에 대한 인증된 요청을 처리합니다.

   create-resource-server 명령을 사용하여 사용자 풀의 `userPoolId`, 리소스 서버의 고유 식별자 및 이름, 범위, 리소스를 생성할 리전을 지정합니다.

   이 예제에서는 식별자와 이름으로 `dcv-session-manager`를 사용하고 범위 이름 및 설명으로 `sm_scope`를 사용합니다.

   ```
   $ aws cognito-idp create-resource-server --user-pool-id us-east-1_QLEXAMPLE
    --identifier dcv-session-manager --name dcv-session-manager --scopes
    ScopeName=sm_scope,ScopeDescription=sm_scope --region us-east-1
   ```

   출력 예

   ```
   {
       "ResourceServer": {
           "UserPoolId": "us-east-1_QLEXAMPLE",
           "Identifier": "dcv-session-manager",
           "Name": "dcv-session-manager",
           "Scopes": [
           {
               "ScopeName": "sm_scope",
               "ScopeDescription": "sm_scope"
           }]
       }
   }
   ```

5. 사용자 풀 클라이언트를 업데이트합니다.

   update-user-pool-client 명령을 사용합니다. 사용자 풀의 `userPoolId`, 사용자 풀 클라이언트의 `ClientId`, 리전을 지정합니다. `--allowed-o-auth-flows`의 경우 클라이언트 ID와 클라이언트 암호의 조합을 사용하여 클라이언트가 토큰 엔드포인트에서 액세스 토큰을 가져와야 한다는 것을 나타내도록 `client_credentials`를 지정합니다. `--allowed-o-auth-scopes`의 경우 다음과 같이 리소스 서버 식별자와 범위 이름을

resource_server_identifier/*scope_name*과 같이 지정합니다. 클라이언트가 Cognito 사용자 풀과 상호 작용할 때 OAuth 프로토콜을 따르도록 허용된다는 것을 나타내도록 `--allowed-o-auth-flows-user-pool-client`를 포함합니다.

```
$ aws cognito-idp update-user-pool-client --user-pool-id us-east-1_QLEXAMPLE --
client-id 2l9273hp6k2ut5cugg9EXAMPLE --allowed-o-auth-flows client_credentials --
allowed-o-auth-scopes dcv-session-manager/sm_scope --allowed-o-auth-flows-user-
pool-client --region us-east-1
```

출력 예

```
{
    "UserPoolClient": {
        "UserPoolId": "us-east-1_QLEXAMPLE",
        "ClientName": "dcv-session-manager-client-app",
        "ClientId": "2l9273hp6k2ut5cugg9EXAMPLE",
        "ClientSecret": "1vp5e8nec7cbf4m9me55mbmht91u61hlh0a78rq1qki1lEXAMPLE",
        "LastModifiedDate": 1602512103.099,
        "CreationDate": 1602510291.498,
        "RefreshTokenValidity": 30,
        "AllowedOAuthFlows": [
            "client_credentials"
        ],
        "AllowedOAuthScopes": [
            "dcv-session-manager/sm_scope"
        ],
        "AllowedOAuthFlowsUserPoolClient": true
    }
}
```

> **Note**
>
> 이제 사용자 풀에서 액세스 토큰을 제공하고 인증할 준비가 되었습니다. 이 예제에서 권한 부여 서버의 URL은 `https://cognito-idp.`*us-east-1*`.amazonaws.com/`*us-east-1_QLEXAMPLE*`/.well-known/jwks.json`입니다.

6. 구성을 테스트합니다.

```
$ curl -H "Authorization: Basic `echo -
n 2l9273hp6k2ut5cugg9EXAMPLE:1vp5e8nec7cbf4m9me55mbmht91u61hlh0a78rq1qki1lEXAMPLE
```

```
 | base64`" -H "Content-Type: application/x-www-form-urlencoded" -X
 POST "https://mydomain-544fa30f-c0e5-4a02-8d2a-a3761EXAMPLE.auth.us-
east-1.amazoncognito.com/oauth2/token?grant_type=client_credentials&scope=dcv-
session-manager/sm_scope"
```

출력 예

```
{
"access_token":"eyJraWQiOiJGQ0VaRFpJUUptT3NSaW41MmtqaDdEbTZYb0RnSTQ5b2VUT0cxUUI1Q2VJPSIsImF
Zkfi0HIDsd6audjTXKzHlZGScr6ROdZtId5dThkpEZiSx0YwiiWe9crAlqoazlDcCsUJHIXDtgKW64pSj3-
uQQGg1Jv_tyVjhrA4JbD0k67WS2V9NW-
uZ7t4zwwaUmOi3KzpBMi54fpVgPaewiVlUm_aS4LUFcWT6hVJjiZF7om7984qb2gOa14iZxpXPBJTZX_gtG9EtvnS9u
"expires_in":3600,
"token_type":"Bearer"
}
```

7. register-auth-server 명령을 사용하여 브로커와 함께 사용할 외부 권한 부여 서버를 등록합니다.

```
$ sudo -u root dcv-session-manager-broker register-auth-server --url  https://
cognito-idp.us-east-1.amazonaws.com/us-east-1_QLEXAMPLE/.well-known/jwks.json
```

이제 개발자가 서버를 사용하여 액세스 토큰을 요청할 수 있습니다. 액세스 토큰을 요청할 때 여기에 생성된 클라이언트 ID, 클라이언트 암호 및 서버 URL을 제공하세요. 액세스 토큰 요청에 대한 자세한 내용은 NICE DCV 세션 관리자 개발자 안내서의 액세스 토큰 확보 및 API 요청을 참조하세요.

브로커 지속성 구성

세션 관리자 브로커는 외부 데이터베이스와의 통합을 지원합니다. 세션 관리자는 외부 데이터베이스를 통해 상태 데이터와 키를 보관하여 나중에 사용할 수 있습니다. 실제로 브로커 데이터는 클러스터를 통해 분산되므로 호스트를 재부팅해야 하거나 클러스터가 종료될 경우 데이터 손실이 발생하기 쉽습니다. 이 기능을 활성화하면 브로커 노드를 추가 및 제거할 수 있습니다. 또한 키를 재생성하지 않고도, 또는 어떤 NICE DCV 서버가 열려 있거나 닫혀 있는지에 대한 정보를 손실하지 않고도 클러스터를 중지하고 다시 시작할 수 있습니다.

다음 유형의 정보가 지속되도록 설정할 수 있습니다.

- 클라이언트와의 연결을 구성하기 위한 세션을 설정하는 데 사용되는 키
- 전송 중 세션 데이터
- NICE DCV 서버 상태

NICE DCV 세션 관리자는 DynamoDB, MariaDB, MySQL 데이터베이스를 지원합니다. 이 기능을 사용하려면 이러한 데이터베이스 중 하나를 설정하고 관리해야 합니다. 브로커 머신이 Amazon EC2에 호스팅되는 경우 추가 설정이 필요하지 않으므로 DynamoDB를 외부 데이터베이스로 사용하는 것이 좋습니다.

> **Note**
>
> 외부 데이터베이스를 실행할 때 추가 비용이 발생할 수 있습니다. DynamoDB 요금에 대한 자세한 내용은 프로비저닝된 용량 요금을 참조하세요.

DynamoDB에서 지속되도록 브로커를 구성합니다.

DynamoDB에 데이터를 저장하하도록 브로커를 구성합니다.

1. 선호하는 텍스트 편집기를 사용하여 `/etc/dcv-session-manager-broker/session-manager-broker.properties`를 열고 다음과 같이 편집합니다.

   - Set `enable-persistence = true`
   - Set `persistence-db = dynamodb`
   - `dynamodb-region`에 브로커 데이터가 포함된 테이블을 저장할 &aws; 리전을 지정합니다. 지원되는 리전의 목록은 DynamoDB 서비스 엔드포인트를 참조하세요.
   - `dynamodb-table-rcu`에 각 테이블에서 지원하는 읽기 용량 단위(RCU)의 양을 지정합니다. RCU에 대한 자세한 내용은 DynamoDB 프로비저닝 용량을 참조하세요.
   - `dynamodb-table-wcu`에 각 테이블에서 지원하는 쓰기 용량 단위(WCU)의 양을 지정합니다. WCU에 대한 자세한 내용은 DynamoDB 프로비저닝 용량을 참조하세요.
   - dynamodb-table-name-prefix의 경우 각 DynamoDB 테이블에 추가되는 접두사를 지정합니다(동일한 계정을 사용하는 여러 브로커 클러스터를 구분하는 데 유용함). 영숫자와 마침표, 대시, 밑줄 기호만 허용됩니다.
2. 클러스터의 모든 브로커를 중지합니다. 각 브로커에 대해 다음의 명령을 실행합니다.

   ```
   sudo systemctl stop dcv-session-manager-broker
   ```
3. 클러스터의 모든 브로커가 중지되었는지 확인한 다음 모든 브로커를 다시 시작합니다. 다음 명령을 실행하여 각 브로커를 시작합니다.

   ```
   sudo systemctl start dcv-session-manager-broker
   ```

브로커 호스트에는 DynamoDB API를 호출할 권한이 있어야 합니다. Amazon EC2 인스턴스에서 보안 인증 정보는 Amazon EC2 메타데이터 서비스를 사용하여 자동으로 검색됩니다. 다른 보안 인증 정보를 지정해야 하는 경우 지원되는 보안 인증 정보 검색 기술 중 하나(예: Java 시스템 속성 또는 환경 변수)를 사용하여 보안 인증 정보를 설정할 수 있습니다. 자세한 내용은 &aws; 보안 인증 정보 제공 및 검색을 참조하세요.

MariaDB/MySQL에서 계속 작동하도록 브로커를 설정합니다.

> **Note**
>
> `/etc/dcv-session-manager-broker/session-manager-broker.properties` 파일에는 민감한 데이터가 포함되어 있습니다. 기본적으로 쓰기 권한은 루트로 제한되며 읽기 권한은 루트 및 브로커를 실행하는 사용자로 제한됩니다. 기본적으로 `dcvsmbroker` 사용자에게 권한이 주어집니다. 브로커는 시작 시 파일에 필요한 권한이 있는지 확인합니다.

MariaDB/MySQL에서 데이터를 유지하도록 브로커를 구성합니다.

1. 선호하는 텍스트 편집기를 사용하여 `/etc/dcv-session-manager-broker/session-manager-broker.properties`를 열고 다음과 같이 편집합니다.

   - Set `enable-persistence = true`
   - Set `persistence-db = mysql`
   - Set `jdbc-connection-url = jdbc:mysql://<db_endpoint>:<db_port>/<db_name>?createDatabaseIfNotExist=true`

     이 구성에서 <db_endpoint>는 데이터베이스 엔드포인트, <db_port>는 데이터베이스 포트, <db_name>은 데이터베이스 이름입니다.
   - `jdbc-user`에 데이터베이스에 액세스할 수 있는 사용자 이름을 지정합니다.
   - `jdbc-password`에 데이터베이스에 액세스할 수 있는 사용자 암호를 지정합니다.
2. 클러스터의 모든 브로커를 중지합니다. 각 브로커에 대해 다음의 명령을 실행합니다.

   ```
   sudo systemctl stop dcv-session-manager-broker
   ```
3. 클러스터의 모든 브로커가 중지되었는지 확인한 다음 모든 브로커를 다시 시작합니다. 각 브로커에 대해 다음의 명령을 실행합니다.

```
sudo systemctl start dcv-session-manager-broker
```

NICE DCV 연결 게이트웨이와 통합

NICE DCV 연결 게이트웨이는 사용자가 LAN 또는 VPC에 대한 단일 액세스 포인트를 통해 NICE DCV 서버 플릿에 액세스할 수 있도록 하는 설치 가능한 소프트웨어 패키지입니다.

인프라에 NICE DCV 연결 게이트웨이를 통해 액세스할 수 있는 NICE DCV 서버가 포함된 경우 NICE DCV 연결 게이트웨이를 통합하도록 세션 관리자를 구성할 수 있습니다. 다음 섹션에 설명된 단계를 따르면 브로커가 연결 게이트웨이의 세션 해석기 역할을 수행합니다. 즉, 브로커가 추가 HTTP 엔드포인트를 표시합니다. 연결 게이트웨이는 엔드포인트에 API를 직접 호출하여 NICE DCV 연결을 브로커가 선택한 호스트로 라우팅하는 데 필요한 정보를 검색합니다.

세션 관리자 브로커를 NICE DCV 연결 게이트웨이의 세션 해석기로 설정

세션 관리자 브로커 측

1. 선호하는 텍스트 편집기를 사용하여 `/etc/dcv-session-manager-broker/session-manager-broker.properties`를 열고 다음 변경 사항을 적용합니다.

   - Set `enable-gateway = true`
   - `gateway-to-broker-connector-https-port`를 사용 가능한 TCP 포트로 설정(기본값 8447)
   - `gateway-to-broker-connector-bind-host`를 NICE DCV 연결 게이트웨이 연결을 위해 브로커가 바인딩하는 호스트의 IP 주소로 설정(기본값 0.0.0.0)
2. 그런 다음 아래 명령을 실행하여 브로커를 중지하고 다시 시작합니다.

   ```
   sudo systemctl stop dcv-session-manager-broker
   ```

   ```
   sudo systemctl start dcv-session-manager-broker
   ```
3. 브로커의 자체 서명 인증서 사본을 검색하여 사용자 디렉터리에 저장합니다.

   ```
   sudo cp /var/lib/dcvsmbroker/security/dcvsmbroker_ca.pem $HOME
   ```

   다음 단계에서 NICE DCV 연결 게이트웨이를 설치할 때 이 사본이 필요합니다.

NICE DCV 연결 게이트웨이 측

- NICE DCV 연결 게이트웨이 설명서의 이 섹션을 따르세요.

  NICE DCV 연결 게이트웨이는 브로커에 HTTP API 직접 호출을 보내므로, 브로커가 자체 서명된 인증서를 사용하는 경우 브로커 인증서를 (이전 단계에서 검색한) NICE DCV 연결 게이트웨이 호스트에 복사하고 NICE DCV 연결 게이트웨이 구성의 `ca-file` 섹션에서 `[resolver]` 파라미터를 설정해야 합니다.

선택 사항 - TLS 클라이언트 인증 활성화

이전 단계를 완료하면 세션 관리자와 연결 게이트웨이가 보안 채널을 통해 통신할 수 있으며, 이 채널에서 연결 게이트웨이는 세션 관리자 브로커의 ID를 확인할 수 있습니다. 보안 채널을 설정하기 전에 세션 관리자 브로커도 연결 게이트웨이의 ID를 확인하도록 하려면 다음 섹션의 단계에 따라 TLS 클라이언트 인증 기능을 활성화해야 합니다.

> **Note**
>
> 세션 관리자가 로드 밸런서 뒤에 있으면 Application Load Balancer(ALB) 또는 Gateway Load Balancer(GLB)와 같이 TLS 연결이 종료되는 로드 밸런서에서 TLS 클라이언트 인증을 활성화할 수 없습니다. Network Load Balancer(NLB)와 같이 TLS 종료가 없는 로드 밸런서만 지원할 수 있습니다. ALB 또는 GLB를 사용하는 경우 특정 보안 그룹만 로드 밸런서에 연결할 수 있도록 하여 보안 수준을 강화할 수 있습니다. 보안 그룹에 대한 자세한 내용은 VPC의 보안 그룹을 참조하세요.

세션 관리자 브로커 측

1. 세션 관리자 브로커와 NICE DCV 연결 게이트웨이 간의 통신을 위한 TLS 클라이언트 인증을 활성화하려면 다음 단계를 따르세요.
2. 다음을 실행하여 필요한 키와 인증서를 생성합니다. 명령의 출력은 보안 인증이 생성된 폴더와 TrustStore 파일을 만드는 데 사용된 암호를 알려줍니다.

   ```
   sudo /usr/share/dcv-session-manager-broker/bin/gen-gateway-certificates.sh
   ```

3. NICE DCV 연결 게이트웨이의 프라이빗 키 및 자체 서명 인증서 사본을 사용자 디렉터리에 저장합니다. 다음 단계에서 NICE DCV 연결 게이트웨이에 대해 TLS 클라이언트 인증을 활성화할 때 이 사본이 필요합니다.

```
sudo cp /etc/dcv-session-manager-broker/resolver-creds/dcv_gateway_key.pem $HOME
```

```
sudo cp /etc/dcv-session-manager-broker/resolver-creds/dcv_gateway_cert.pem $HOME
```

4. 그런 다음 선호하는 텍스트 편집기를 사용하여 /etc/dcv-session-manager-broker/session-manager-broker.properties를 열고 다음을 수행합니다.

   - `enable-tls-client-auth-gateway`를 `true`로 설정합니다.
   - `gateway-to-broker-connector-trust-store-file`을 이전 단계에서 생성한 TrustStore 파일의 경로로 설정합니다.
   - `gateway-to-broker-connector-trust-store-pass`를 이전 단계에서 TrustStore 파일을 생성할 때 사용되는 암호로 설정합니다.

5. 그런 다음 아래 명령을 실행하여 브로커를 중지하고 다시 시작합니다.

```
sudo systemctl stop dcv-session-manager-broker
```

```
sudo systemctl start dcv-session-manager-broker
```

NICE DCV 연결 게이트웨이 측

- NICE DCV 연결 게이트웨이 설명서의 이 섹션을 따르세요.
  - `[resolver]` 섹션에서 `cert-file` 파라미터를 설정할 때는 이전 단계에서 복사한 인증서 파일의 전체 경로를 사용하세요.
  - `[resolver]` 섹션에서 `cert-key-file` 파라미터를 설정할 때는 이전 단계에서 복사한 키 파일의 전체 경로를 사용하세요.

NICE DCV 세션 관리자 NICE DCV 서버 - DNS 매핑

NICE DCV 연결 게이트웨이를 DCV 서버 인스턴스에 연결하려면 NICE DCV 서버의 DNS 이름이 필요합니다. 이 섹션에서는 각 DCV 서버와 관련 DNS 이름 간의 매핑이 포함된 JSON 파일을 정의하는 방법을 설명합니다.

파일 구조

매핑은 다음 필드가 있는 JSON 객체 목록으로 구성됩니다.

```
[
 {
  “ServerIdType”: "Ip",
  “ServerId”: "192.168.0.1",
  “DnsNames”:
  {
   “InternalDnsName”: "internal"
  }
 },
 ...
]
```

여기서 각 항목은 다음과 같습니다.

`ServerIdType:`

값이 참조하는 ID 유형을 식별합니다. 현재 사용 가능한 값은 ipAddress, agentServerId, instanceId입니다.

`Ip:`

Amazon EC2와 온프레미스 인프라 모두에서 사용할 수 있으며, 시스템 관리자가 ifconfig(Linux) 또는 ipconfig(Windows) 명령을 사용하여 빠르게 검색할 수 있습니다. 이 정보는 DescribeServers API 응답에서도 확인할 수 있습니다.

`Id:`

Amazon EC2와 온프레미스 인프라 모두에서 사용할 수 있으며, 세션 관리자 에이전트는 호스트 이름 또는 IP 주소가 변경될 때마다 새 UUID를 생성합니다. 이 정보는 DescribeServers API 응답에서 확인할 수 있습니다.

`Host.Aws.Ec2InstanceId:`

Amazon EC2 인스턴스에만 사용할 수 있으며, 시스템을 고유하게 식별하며 인스턴스 재부팅 후에도 변경되지 않습니다. http://169.254.169.254/latest/meta-data/instance-id에 연결하여 호스트에서 검색할 수 있습니다. 이 정보는 DescribeServers API 응답에서도 확인할 수 있습니다.

`ServerId:`

네트워크에서 각 NICE DCV 서버를 고유하게 식별하는 지정된 유형의 ID입니다.

`DnsNames:`

NICE DCV 서버와 관련된 DNS 이름을 포함하는 개체로, 이 개체에는 다음이 포함됩니다.

`InternalDnsNames:`

NICE DCV 연결 게이트웨이가 인스턴스에 연결할 때 사용하는 DNS 이름입니다.

세션 관리자 브로커 CLI 명령 `register-server-dns-mapping`을 사용하여 파일에서 매핑을 로드하고(명령 페이지 참조: register-server-dns-mapping), `describe-server-dns-mappings`를 사용하여 세션 관리자 브로커에 현재 로드된 매핑을 나열하세요(명령 페이지 참조: describe-server-dns-mappings).

지속성

여러 브로커 또는 전체 클러스터가 다운될 때 매핑 손실을 방지할 수 있도록 세션 관리자 브로커의 지속성 기능을 활성화하는 것이 적극 권장됩니다. 데이터 지속성 활성화에 대한 자세한 내용은 브로커 지속성 구성을 참조하세요.

Amazon CloudWatch와 통합

세션 관리자는 Amazon EC2 인스턴스에서 실행되는 브로커와 온프레미스 호스트에서 실행되는 브로커의 Amazon CloudWatch 통합을 지원합니다.

Amazon CloudWatch는 Amazon Web Services(AWS) 리소스 및 AWS에서 실행되는 애플리케이션을 실시간으로 모니터링합니다. CloudWatch를 사용하여 리소스 및 애플리케이션에 대해 측정할 수 있는 변수인 지표를 수집하고 추적할 수 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하십시오.

Amazon CloudWatch에 다음 지표 데이터를 전송하도록 세션 관리자 브로커를 구성할 수 있습니다.

- `Number of DCV servers`—브로커가 관리하는 DCV 서버의 수
- `Number of ready DCV servers`—브로커가 관리하는 `READY` 상태의 DCV 서버 수
- `Number of DCV sessions`—브로커가 관리하는 DCV 세션의 수
- `Number of DCV console sessions`—브로커가 관리하는 DCV 콘솔 세션의 수
- `Number of DCV virtual sessions`—브로커가 관리하는 DCV 가상 세션의 수
- `Heap memory used`—브로커가 사용하는 힙 메모리의 양
- `Off-heap memory used`—브로커가 사용하는 오프 힙 메모리의 양

- `Describe sessions request time`—DescribeSessions API 요청을 완료하는 데 소요된 시간입니다.
- `Delete sessions request time`—DeleteSessions API 요청을 완료하는 데 소요된 시간입니다.
- `Create sessions request time`—CreateSessions API 요청을 완료하는 데 소요된 시간입니다.
- `Get session connection data request time`—GetSessionConnectionData Data API 요청을 완료하는 데 걸린 시간입니다.
- `Update session permissions sequest time`—UpdateSessionPermissions API 요청을 완료하는 데 걸린 시간입니다.

브로커가 지표 데이터를 Amazon CloudWatch로 전송하도록 구성하려면

1. 선호하는 텍스트 편집기를 사용하여 `/etc/dcv-session-manager-broker/session-manager-broker.properties`를 열고 다음과 같이 진행합니다.

   - `enable-cloud-watch-metrics`를 `true`로 설정합니다.
   - `cloud-watch-region`의 경우 지표 데이터를 수집할 리전을 지정합니다.

   > **Note**
   >
   > Amazon EC2 인스턴스에서 브로커를 실행 중인 경우 이 파라미터는 선택 사항입니다. 이 리전은 인스턴스 메타데이터 서비스(IMDS)에서 자동으로 검색됩니다. 온프레미스 호스트에서 브로커를 실행 중인 경우 이 파라미터는 필수입니다.

2. 브로커를 중지한 후 다시 시작합니다.

```
$  sudo systemctl stop dcv-session-manager-broker
```

```
$  sudo systemctl start dcv-session-manager-broker
```

브로커 호스트에는 `cloudwatch:PutMetricData` API를 호출할 권한도 있어야 합니다. 지원되는 보안 인증 정보 검색 기술 중 하나를 사용하여 AWS 보안 인증 정보를 검색할 수 있습니다. 자세한 내용은 AWS 보안 인증 정보 제공 및 검색을 참조하세요.

NICE DCV 세션 관리자 업그레이드

다음 주제에서는 세션 관리자를 업그레이드하는 방법에 대해 설명합니다.

> **Note**
>
> 새 기능이 도입될 경우 호환성 문제가 발생하지 않도록 세션 관리자 브로커를 업그레이드하기 전에 모든 세션 관리자 에이전트를 업그레이드하는 것이 좋습니다.

주제

NICE DCV 세션 관리자 에이전트 업그레이드

Linux host

> **Note**
>
> 다음은 64비트 x86 호스트에 에이전트를 설치하기 위한 안내입니다. 64비트 ARM 호스트에 에이전트를 설치하려면 Amazon Linux, RHEL, Centos의 경우 *x86_64*를 `aarch64`로 대체하고, Ubuntu의 경우 *amd64*를 `arm64`로 대체합니다.

Linux 호스트에서 에이전트를 업데이트하려면 다음과 같이 하세요.

1. 다음 명령을 실행하여 에이전트를 중지합니다.

```
$  sudo systemctl stop dcv-session-manager-agent
```

2. 설치 패키지를 다운로드합니다.

   - Amazon Linux 2, RHEL 7.x, CentOS 7.x

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerAgents/
nice-dcv-session-manager-agent-2023.1.732-1.el7.x86_64.rpm
```

- RHEL 8.x, CentOS Stream 8, Rocky Linux 8.x

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerAgents/
nice-dcv-session-manager-agent-2023.1.732-1.el8.x86_64.rpm
```

- Ubuntu 20.04

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerAgents/
nice-dcv-session-manager-agent_2023.1.732-1_amd64.ubuntu2004.deb
```

- SUSE Linux Enterprise 12

```
$  curl -O https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerAgents/
nice-dcv-session-manager-agent-2023.1.732-1.sles12.x86_64.rpm
```

- SUSE Linux Enterprise 15

```
$  curl -O https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerAgents/
nice-dcv-session-manager-agent-2023.1.732-1.sles15.x86_64.rpm
```

3. 패키지를 설치합니다.

- Amazon Linux 2, RHEL 7.x, CentOS 7.x

```
$  sudo yum install -y nice-dcv-session-manager-
agent-2023.1.732-1.el7.x86_64.rpm
```

- RHEL 8.x, CentOS Stream 8, Rocky Linux 8.x

```
$  sudo yum install -y nice-dcv-session-manager-
agent-2023.1.732-1.el8.x86_64.rpm
```

- Ubuntu 20.04

```
$  sudo apt install ./nice-dcv-session-manager-
agent_2023.1.732-1_amd64.ubuntu2004.deb
```

- SUSE Linux Enterprise 12

```
$  sudo zypper install nice-dcv-session-manager-
agent-2023.1.732-1.sles12.x86_64.rpm
```

- SUSE Linux Enterprise 15

```
$  sudo zypper install nice-dcv-session-manager-
agent-2023.1.732-1.sles15.x86_64.rpm
```

4. 다음 명령을 실행하여 에이전트를 시작합니다.

```
$  sudo systemctl start dcv-session-manager-agent
```

Windows host

Windows 호스트에서 에이전트를 업데이트하려면 다음과 같이 하세요.

1. 에이전트 서비스를 중지합니다. 명령 프롬프트에서 다음 명령을 실행합니다.

```
C:\>  sc start DcvSessionManagerAgentService
```

2. 에이전트 설치 프로그램을 다운로드합니다.
3. 설치 관리자를 실행합니다. 시작 화면에서 다음을 선택합니다.
4. EULA 화면에서 라이선스 계약을 주의 깊게 읽어보고, 약관에 동의하는 경우 약관에 동의함을 선택하고 다음을 클릭합니다.
5. 설치를 시작하려면 설치를 클릭합니다.
6. 에이전트 서비스를 다시 시작합니다. 명령 프롬프트에서 다음 명령을 실행합니다.

```
C:\>  sc stop DcvSessionManagerAgentService
```

NICE DCV 세션 관리자 브로커 업그레이드

브로커를 업그레이드하려면 다음과 같이 하세요.

1. 브로커를 업그레이드하려는 호스트에 연결합니다.
2. 브로커 서비스를 중지합니다.

```
$  sudo systemctl stop dcv-session-manager-broker
```

3. 설치 패키지를 다운로드합니다.

- Amazon Linux 2, RHEL 7.x, CentOS 7.x

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerBrokers/nice-dcv-session-manager-broker-2023.1.410-1.el7.noarch.rpm
```

- RHEL 8.x, CentOS Stream 8, Rocky Linux 8.x

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerBrokers/nice-dcv-session-manager-broker-2023.1.410-1.el8.noarch.rpm
```

- Ubuntu 20.04

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerBrokers/nice-dcv-session-manager-broker-2023.1.410-1_all.ubuntu2004.deb
```

4. 패키지를 설치합니다.

- Amazon Linux 2, RHEL 7.x, CentOS 7.x

```
$  sudo yum install -y nice-dcv-session-manager-broker-2023.1.410-1.el7.noarch.rpm
```

- RHEL 8.x, CentOS Stream 8, Rocky Linux 8.x

```
$  sudo yum install -y nice-dcv-session-manager-broker-2023.1.410-1.el8.noarch.rpm
```

- Ubuntu 20.04

```
$  sudo apt install -y nice-dcv-session-manager-broker-2023.1.410-1_all.ubuntu2004.deb
```

5. 브로커 서비스를 시작하고 인스턴스가 시작될 때마다 서비스가 자동으로 시작되는지 확인합니다.

```
$  sudo systemctl start dcv-session-manager-broker && sudo systemctl enable dcv-session-manager-broker
```

브로커 CLI 참조

이 섹션에서는 브로커 명령줄 인터페이스(CLI)를 사용하는 방법을 설명합니다.

외부 인증 서버를 사용하여 OAuth 2.0 액세스 토큰을 생성하는 경우 다음 명령을 사용하세요.

- register-auth-server
- list-auth-servers
- unregister-auth-server

세션 관리자 브로커를 OAuth 2.0 인증 서버로 사용하는 경우 다음 명령을 사용하세요.

- register-api-client
- describe-api-clients
- unregister-api-client
- renew-auth-server-api-key

다음 명령을 사용하여 세션 관리자 에이전트를 관리할 수 있습니다.

- generate-software-statement
- describe-software-statements
- deactivate-software-statement
- describe-agent-clients
- unregister-agent-client

DCV 서버 - DNS 이름 매핑 파일을 관리하려면 다음 명령을 사용하세요.

- register-server-dns-mappings
- describe-server-dns-mappings

register-auth-server

브로커와 함께 사용할 외부 인증 서버를 등록합니다.

기본적으로 세션 관리자는 브로커를 인증 서버로 사용하여 OAuth 2.0 액세스 토큰을 생성합니다. 브로커를 인증 서버로 사용하는 경우 추가 구성이 필요하지 않습니다.

하지만 Active Directory 또는 Amazon Cognito와 같은 외부 인증 서버를 사용하기로 선택한 경우 이 명령을 사용하여 외부 인증 서버를 등록해야 합니다.

주제

조건

```
sudo -u root dcv-session-manager-broker register-auth-server --url  server_url.well-known/jwks.json
```

옵션

`--url`

사용할 외부 인증 서버의 URL입니다. `.well-known/jwks.json`을 인증 서버 URL에 추가해야 합니다.

유형: 문자열

필수 항목 여부: 예

예

다음 예제에서는 URL이 `https://my-auth-server.com/`인 외부 인증 서버를 등록합니다.

명령

```
sudo -u root dcv-session-manager-broker register-auth-server --url https://my-auth-server.com/.well-known/jwks.json
```

출력

```
Jwk url registered.
```

list-auth-servers

등록된 외부 인증 서버를 나열합니다.

주제

조건

```
sudo -u root dcv-session-manager-broker list-auth-servers
```

출력

`Urls`

등록된 외부 인증 서버의 URL입니다.

예

다음 예제에서는 등록된 모든 외부 인증 서버를 나열합니다.

명령

```
sudo -u root dcv-session-manager-broker list-auth-servers
```

출력

```
Urls: [ "https://my-auth-server.com/.well-known/jwks.json" ]
```

unregister-auth-server

외부 인증 서버의 등록을 취소합니다. 외부 인증 서버를 등록 취소한 후에는 OAuth 2.0 액세스 토큰을 생성하는 데 더 이상 해당 서버를 사용할 수 없습니다.

주제

조건

```
sudo -u root dcv-session-manager-broker unregister-auth-server --url  server_url.well-known/jwks.json
```

옵션

`--url`

등록 취소할 외부 인증 서버의 URL입니다. `.well-known/jwks.json`을 인증 서버 URL에 추가해야 합니다.

유형: 문자열

필수 항목 여부: 예

출력

`Url`

등록되지 않은 외부 인증 서버의 URL입니다.

예

다음 예제에서는 URL이 `https://my-auth-server.com/`인 외부 인증 서버를 등록합니다.

명령

```
sudo -u root dcv-session-manager-broker unregister-auth-server --url https://my-auth-server.com/.well-known/jwks.json
```

출력

```
Jwk urlhttps://my-auth-server.com/.well-known/jwks.json unregistered
```

register-api-client

세션 관리자 클라이언트를 브로커에 등록하고 클라이언트가 API 요청에 필요한 OAuth 2.0 액세스 토큰을 검색하는 데 사용할 수 있는 클라이언트 보안 인증 정보를 생성합니다.

> **Important**
>
> 보안 인증 정보를 안전한 위치에 저장합니다. 이 정보는 나중에 복구할 수 없습니다.

이 명령은 브로커가 OAuth 2.0 인증 서버로 사용되는 경우에만 사용됩니다.

주제

조건

```
sudo -u root dcv-session-manager-broker register-api-client --client-name client_name
```

옵션

`--name`

세션 관리자 클라이언트를 식별하는 데 사용되는 고유한 이름입니다.

유형: 문자열

필수 항목 여부: 예

출력

`client-id`

세션 관리자 클라이언트가 OAuth 2.0 액세스 토큰을 검색하는 데 사용하는 고유한 클라이언트 ID입니다.

`client-password`

세션 관리자 클라이언트가 OAuth 2.0 액세스 토큰을 검색하는 데 사용하는 암호입니다.

예

다음 예제에서는 이름이 `my-sm-client`인 클라이언트를 등록합니다.

명령

```
sudo -u root dcv-session-manager-broker register-api-client --client-name my-sm-client
```

출력

```
client-id: 21cfe9cf-61d7-4c53-b1b6-cf248EXAMPLE
client-password: NjVmZDRlN2ItNjNmYS00M2QxLWFlZmMtZmNmMDNkMEXAMPLE
```

describe-api-clients

브로커에 등록된 세션 관리자 클라이언트를 나열합니다.

주제

조건

```
sudo -u root dcv-session-manager-broker describe-api-clients
```

출력

name

세션 관리자 클라이언트의 고유한 이름입니다.

id

세션 관리자 클라이언트의 고유한 ID입니다.

active

세션 관리자 클라이언트의 상태를 나타냅니다. 클라이언트가 활성 상태인 경우 값은 `true`이고, 아니면 `false`입니다.

예

다음 예제에서는 등록된 세션 관리자 클라이언트를 나열합니다.

명령

```
sudo -u root dcv-session-manager-broker describe-api-clients
```

출력

```
Api clients
[ {
"name" : "client-abc",
"id" : "f855b54b-40d4-4769-b792-b727bEXAMPLE",
"active" : false
}, {
"name" : "client-xyz",
"id" : "21cfe9cf-61d7-4c53-b1b6-cf248EXAMPLE",
"active" : true
}]
```

unregister-api-client

등록된 세션 관리자 클라이언트를 비활성화합니다. 비활성화된 세션 관리자 클라이언트는 더 이상 보안 인증 정보를 사용하여 OAuth 2.0 액세스 토큰을 검색할 수 없습니다.

주제

조건

```
sudo -u root dcv-session-manager-broker unregister-api-client --client-id client_id
```

옵션

`--client -id`

비활성화할 세션 관리자 클라이언트의 클라이언트 ID입니다.

유형: 문자열

필수 항목 여부: 예

예

다음 예제에서는 클라이언트 ID가 `f855b54b-40d4-4769-b792-b727bEXAMPLE`인 세션 관리자 클라이언트를 비활성화합니다.

명령

```
sudo -u root dcv-session-manager-broker unregister-api-client --client-id
 f855b54b-40d4-4769-b792-b727bEXAMPLE
```

출력

```
Client f855b54b-40d4-4769-b792-b727bEXAMPLE unregistered.
```

renew-auth-server-api-key

브로커가 세션 관리자 클라이언트에 제공되는 OAuth 2.0 액세스 토큰에 서명하는 데 사용하는 퍼블릭 키와 프라이빗 키를 갱신합니다. 키를 갱신하는 경우 API 요청에 필요한 새 프라이빗 키를 개발자에게 제공해야 합니다.

주제

조건

```
sudo -u root dcv-session-manager-broker renew-auth-server-api-key
```

예

다음 예제에서는 퍼블릭 키와 프라이빗 키를 갱신합니다.

명령

```
sudo -u root dcv-session-manager-broker renew-auth-server-api-key
```

출력

```
Keys renewed.
```

generate-software-statement

소프트웨어 명령문을 생성합니다.

에이전트는 브로커에 등록되어 있어야 통신이 가능합니다. 에이전트를 브로커에 등록하려면 소프트웨어 명령문이 필요합니다. 에이전트에 소프트웨어 명령문이 있으면 OAuth 2.0 동적 클라이언트 등록 프로토콜을 사용하여 브로커에 자동으로 등록할 수 있습니다. 에이전트가 브로커에 등록되면 브로커 인증에 사용하는 클라이언트 ID와 클라이언트 암호가 주어집니다.

브로커와 에이전트는 처음 설치될 때 기본 소프트웨어 명령문을 받아 사용합니다. 기본 소프트웨어 명령문을 계속 사용해도 되고, 새 소프트웨어 명령문을 생성할 수도 있습니다. 새 소프트웨어 명령문을

생성하는 경우 소프트웨어 명령문을 에이전트의 새 파일에 넣은 다음, 파일 경로를 agent.conf 파일의 agent.software_statement_path 파라미터에 추가해야 합니다. 이 작업을 완료한 후에는 에이전트가 새 소프트웨어 명령문을 사용하여 브로커에 등록할 수 있도록 에이전트를 중지했다가 다시 시작하세요.

주제

조건

```
sudo -u root dcv-session-manager-broker generate-software-statement
```

출력

software-statement

소프트웨어 명령문입니다.

예

다음 예제는 소프트웨어 명령문을 생성합니다.

명령

```
sudo -u root dcv-session-manager-broker generate-software-statement
```

출력

```
software-statement:
 ewogICJpZCIgOiAiYjc1NTVhN2QtNWI0MC00OTJhLWJjOTUtNmUzOWNhYzkxMDcxIiwKICAiYWN0aXZlIiA6IHRydWUsCi
```

describe-software-statements

기존 소프트웨어 명령문을 설명합니다.

주제

조건

```
sudo -u root dcv-session-manager-broker describe-software-statements
```

출력

software-statement

소프트웨어 명령문입니다.

issued-at

소프트웨어가 생성된 날짜 및 시간입니다.

is-active

소프트웨어 명령문의 현재 상태입니다. 소프트웨어 명령문이 활성화되어 있는 경우 `true`, 그렇지 않으면 `false`입니다.

예

다음 예제는 소프트웨어 명령문을 생성합니다.

명령

```
sudo -u root dcv-session-manager-broker describe-software-statements
```

출력

```
Software Statements
[ {
"software-statement" :
 "ewogICJpZCIgOiAiYmEEXAMPLEYtNzUwNy00YmFhLTliZWItYTA1MmJjZTE3NDJjIiwKICAiaXNzdWVkQXQiIDogMTU5N
```

```
"issued-at" : "2020.08.05 15:38:32 +0000",
"is-active" : "true"
}, {
"software-statement" :
 "EXAMPLEpZCIgOiAiYjc1NTVhN2QtNWI0MC00OTJhLWJjOTUtNmUzOWNhYzkxMDcxIiwKICAiaXNzdWEXAMPLEDogMTU5N
"issued-at" : "2020.08.07 10:24:41 +0000",
"is-active" : "true"
} ]
```

deactivate-software-statement

소프트웨어 명령문을 비활성화합니다. 소프트웨어 명령문을 비활성화하면 에이전트 등록에 해당 명령문을 더 이상 사용할 수 없습니다.

주제

조건

```
sudo -u root dcv-session-manager-broker deactivate-software-statement --software-
statement software_statement
```

옵션

`--software-statement`

비활성화할 소프트웨어 명령문입니다.

유형: 문자열

필수 항목 여부: 예

예

다음 예제는 소프트웨어 명령문을 비활성화합니다.

명령

```
sudo -u root dcv-session-manager-broker deactivate-software-statement --software-
statement
 EXAMPLEpZCIgOiAiYjc1NTVhN2QtNWI0MC00OTJhLWJjOTUtNmUzOWNhYzkxMDcxIiwKICAiaXNEXAMPLEQiIDogMTU5Nj
```

출력

```
Software statement
 EXAMPLEpZCIgOiAiYjc1NTVhN2QtNWI0MC00OTJhLWJjOTUtNmUzOWNhYzkxMDcxIiwKICAiaXNEXAMPLEQiIDogMTU5Nj
 deactivated
```

describe-agent-clients

브로커에 등록된 에이전트에 대해 설명합니다.

주제

조건

```
sudo -u root dcv-session-manager-broker describe-agent-clients
```

출력

name

에이전트의 이름입니다.

id

에이전트의 고유 ID입니다.

active

에이전트의 상태입니다. 에이전트가 활성 상태인 경우 `true`, 그렇지 않으면 `false`입니다.

예

다음 예제에서는 에이전트를 설명합니다.

명령

```
sudo -u root dcv-session-manager-broker describe-agent-clients
```

출력

```
Session manager agent clients
[ {
"name" : "test",
"id" : "6bc05632-70cb-4410-9e54-eaf9bEXAMPLE",
"active" : true
}, {
"name" : "test",
"id" : "27131cc2-4c71-4157-a4ca-bde38EXAMPLE",
"active" : true
}, {
"name" : "test",
"id" : "308dd275-2b66-443f-95af-33f63EXAMPLE",
"active" : false
}, {
"name" : "test",
"id" : "ce412d1b-d75c-4510-a11b-9d9a3EXAMPLE",
"active" : true
} ]
```

unregister-agent-client

브로커에서 에이전트 등록을 취소합니다.

주제

조건

```
sudo -u root dcv-session-manager-broker unregister-agent-client --client-id client_id
```

옵션

`--client-id`

등록 취소할 에이전트의 ID입니다.

유형: 문자열

필수 항목 여부: 예

예

다음 예제에서는 에이전트의 등록을 취소합니다.

명령

```
sudo -u root dcv-session-manager-broker unregister-agent-client --client-id
 3b0d7b1d-78c7-4e79-b2e1-b976dEXAMPLE
```

출력

```
Agent client 3b0d7b1d-78c7-4e79-b2e1-b976dEXAMPLE unregistered
```

register-server-dns-mappings

JSON 파일에서 가져온 DCV 서버 - DNS 이름 매핑을 등록합니다.

조건

```
sudo -u root dcv-session-manager-broker register-server-dns-mappings --file-
path file_path
```

옵션

`--file-path`

DCV 서버 - DNS 이름 매핑이 포함된 파일의 경로입니다.

유형: 문자열

필수 항목 여부: 예

예

다음 예제에서는 /tmp/mappings.json 파일에서 DCV 서버 - DNS 이름 매핑을 등록합니다.

명령

```
sudo -u root dcv-session-manager-broker register-server-dns-mappings --file-path /tmp/
mappings.json
```

출력

```
Successfully loaded 2 server id - dns name mappings from file /tmp/mappings.json
```

describe-server-dns-mappings

현재 사용 가능한 DCV 서버 - DNS 이름 매핑을 설명합니다.

조건

```
sudo -u root dcv-session-manager-broker describe-server-dns-mappings
```

출력

`serverIdType`

서버 ID의 유형입니다.

`serverId`

서버의 고유 ID입니다.

dnsNames

내부 및 외부 DNS 이름입니다.

internalDnsNames

내부 DNS 이름입니다.

externalDnsNames

외부 DNS 이름입니다.

예

다음 예제에서는 등록된 DCV 서버 - DNS 이름 매핑을 나열합니다.

명령

```
sudo -u root dcv-session-manager-broker describe-server-dns-mappings
```

출력

```
[
{
 "serverIdType" : "Id",
 "serverId" : "192.168.0.1",
 "dnsNames" : {
  "internalDnsName" : "internal1",
  "externalDnsName" : "external1"
 }
},
{
 "serverIdType" : "Host.Aws.Ec2InstanceId",
 "serverId" : "i-0648aee30bc78bdff",
 "dnsNames" : {
  "internalDnsName" : "internal2",
  "externalDnsName" : "external2"
 }
}
]
```

구성 파일 참조

이 섹션에서는 에이전트 및 브로커 구성 파일에 관한 정보를 제공합니다.

주제

브로커 구성 파일

브로커 구성 파일(`/etc/dcv-session-manager-broker/session-manager-broker.properties`)에는 세션 관리자 기능을 사용자 지정하도록 구성할 수 있는 파라미터가 포함되어 있습니다. 원하는 텍스트 편집기를 사용하여 구성 파일을 편집할 수 있습니다.

> **Note**
>
> `/etc/dcv-session-manager-broker/session-manager-broker.properties` 파일에는 민감한 데이터가 포함되어 있습니다. 기본적으로 쓰기 권한은 루트로 제한되며 읽기 권한은 루트 및 브로커를 실행하는 사용자로 제한됩니다. 기본적으로 `dcvsmbroker` 사용자에게 권한이 주어집니다. 브로커는 시작 시 파일에 필요한 권한이 있는지 확인합니다.

다음 표에는 브로커 구성 파일의 파라미터가 나열되어 있습니다.

파라미터 이름	필수	기본값	설명
`broker-java-home`	아니요		브로커가 시스템 기본 디렉터리 대신 사용할 Java 홈 디렉터리의 경로를 지정합니다. 설정하면 브로커에서 시작 시 `<broker-java-home>/bin/java` 를 사용합니다.

파라미터 이름	필수	기본값	설명
			팁: 브로커는 Java Runtime Environment 11이 필요하며, 설치 성공 시 종속 항목으로 누락된 경우 설치됩니다. 버전 11이 기본 Java 환경으로 설정되지 않은 경우 다음 명령을 사용하여 해당 홈 디렉터리를 가져올 수 있습니다. `$ sudo alternatives --display java`
sessions creenshot-max-width	아니요	160	GetSessionScreenshots API를 사용하여 캡처한 세션 스크린샷의 최대 너비(픽셀 단위)를 지정합니다.
sessions creenshot-max-height	아니요	100	GetSessionScreenshots API를 사용하여 캡처한 세션 스크린샷의 최대 높이(픽셀 단위)를 지정합니다.

파라미터 이름	필수	기본값	설명
sessionscreenshot-format	아니요	png	GetSessionScreenshots API를 사용하여 캡처한 세션 스크린샷의 이미지 파일 형식입니다.
create-sessions-queue-max-size	아니요	1000	처리되지 않은 CreateSessions API 요청 중 대기열에 넣을 수 있는 요청의 최대 수입니다. 대기열이 가득 차면 처리되지 않은 새 요청이 거부됩니다.
create-sessions-queue-max-time-seconds	아니요	1800	처리되지 않은 CreateSessions API 요청이 대기열에 남아 있을 수 있는 최대 시간(초)입니다. 지정된 시간 내에 요청을 이행할 수 없는 경우 요청이 실패합니다.
sessionmanager-working-path	예	/tmp	브로커가 운영에 필요한 파일을 기록하는 디렉터리의 경로를 지정합니다. 이 디렉터리에는 브로커만 액세스할 수 있어야 합니다.

파라미터 이름	필수	기본값	설명
enable-authorization-server	예	true	브로커가 클라이언트 API용 OAuth 2.0 액세스 토큰을 생성하는 데 사용되는 인증 서버인지 여부를 지정합니다.
enable-authorization	예	true	클라이언트 권한 부여를 활성화하거나 비활성화합니다. 클라이언트 권한 부여를 활성화하는 경우 API 요청 시 클라이언트 API가 액세스 토큰을 제공해야 합니다. 클라이언트 권한 부여를 비활성화하면 클라이언트 API가 액세스 토큰 없이 요청을 할 수 있습니다.
enable-agent-authorizatic	예	true	에이전트 권한 부여를 활성화 또는 비활성화합니다. 에이전트 권한 부여를 활성화한 경우 에이전트는 브로커와 통신할 때 액세스 토큰을 제공해야 합니다.
delete-session-duration-hours	아니요	1	삭제된 세션이 보이지 않고 DescribeSession API 호출에 의해 더 이상 반환되지 않는 시간을 지정합니다.

파라미터 이름	필수	기본값	설명
connect-session-token-duration-minutes	아니요	60	ConnectSession 토큰이 유효한 상태로 유지되는 시간(분)을 지정합니다.
client-to-broker-connector-https-port	예	8443	브로커가 클라이언트 연결을 수신하는 HTTPS 포트를 지정합니다.
client-to-broker-connector-bind-host	아니요	0.0.0.0	브로커가 클라이언트 연결을 위해 바인딩되는 호스트의 IP 주소를 지정합니다.

파라미터 이름	필수	기본값	설명
client-to-broker-connect key-store-file	예		TLS 클라이언트 연결에 사용되는 키 스토어를 지정합니다.
client-to-broker-connect key-store-pass	예		키 스토어 패스를 지정합니다.
agent-to-broker-connector https-port	예	8445	브로커가 에이전트 연결을 수신하는 HTTPS 포트를 지정합니다.

파라미터 이름	필수	기본값	설명
agent-to-broker-connector-bind-host	아니요	0.0.0.0	브로커가 에이전트 연결을 위해 바인딩되는 호스트의 IP 주소를 지정합니다.
agent-to-broker-connector-key-store-file	예		TLS 에이전트 연결에 사용되는 키 스토어를 지정합니다.
agent-to-broker-connector-key-store-pass	예		키 스토어 패스를 지정합니다.
broker-to-broker-port	예	47100	브로커 간 연결에 사용되는 포트를 지정합니다.

파라미터 이름	필수	기본값	설명
broker-to-broker-bind-host	아니요	0.0.0.0	브로커가 브로커 간 연결을 위해 바인딩되는 호스트의 IP 주소를 지정합니다.
broker-to-broker-discovery-port	예	47500	브로커가 서로를 검색하는 데 사용하는 포트를 지정합니다.
broker-to-broker-discovery-address	아니요		플릿에 있는 다른 브로커의 IP 주소와 포트를 *ip_address* :*port* 형식으로 지정합니다. 브로커가 여러 개 있는 경우 값을 쉼표로 구분하세요. broker-to-broker-discovery-multicast-group , broker-to-broker-discovery-multicast-port , broker-to-broker-discovery-AWS-region 또는 broker-to-broker-discovery-AWS-alb-target-group-arn 을 지정하는 경우 이 파라미터를 생략합니다.

파라미터 이름	필수	기본값	설명
broker-to-broker-discovery-multicast-group	아니요		브로커 간 검색을 위한 멀티캐스트 그룹을 지정합니다. broker-to-broker-discovery-addresses , broker-to-broker-discovery-aws-region 또는 broker-to-broker-discovery-AWS-alb-target-group-arn 을 지정하는 경우 이 파라미터를 생략합니다.
broker-to-broker-discovery-multicast-port	아니요		브로커 간 검색을 위한 멀티캐스트 포트를 지정합니다. broker-to-broker-discovery-addresses , broker-to-broker-discovery-AWS-region 또는 broker-to-broker-discovery-AWS-alb-target-group-arn 을 지정하는 경우 이 파라미터를 생략합니다.

파라미터 이름	필수	기본값	설명
broker-to-broker-discovery-AWS-region	아니요		브로커 간 검색에 사용되는 Application Load Balancer의 AWS 리전을 지정합니다. broker-to-broker-discovery-multicast-group , broker-to-broker-discovery-multicast-port 또는 broker-to-broker-discovery-addresses 를 지정하는 경우 이 파라미터를 생략합니다.
broker-to-broker-discovery-AWS-alb-target-group-arn	아니요		브로커 간 검색을 위한 Application Load Balancer 대상 그룹 사용자의 ARN입니다. broker-to-broker-discovery-multicast-group , broker-to-broker-discovery-multicast-port 또는 broker-to-broker-discovery-addresses 를 지정하는 경우 이 파라미터를 생략합니다.

파라미터 이름	필수	기본값	설명
broker-to-broker-distributed-memory-max-size-mb	아니요	4096	단일 브로커가 NICE DCV 세션 데이터를 저장하는 데 사용할 오프 힙 메모리의 최대 양을 지정합니다.
broker-to-broker-key-store-file	예		TLS 브로커 연결에 사용되는 키 스토어를 지정합니다.
broker-to-broker-key-store-pass	예		키 스토어 패스를 지정합니다.
enable-cloud-watch-metrics	아니요	false	Amazon CloudWatch 지표를 활성화하거나 비활성화합니다. CloudWatch 지표를 활성화하는 경우 cloud-watch-region 에 대한 값을 지정해야 할 수 있습니다.

파라미터 이름	필수	기본값	설명
`cloud-watch-region`	아니요	이는 `enable-cloud-watch-metrics` 가 `true`로 설정된 경우에만 필수입니다. 브로커가 Amazon EC2 인스턴스에 설치된 경우 IMDS에서 해당 리전을 검색합니다.	CloudWatch 지표가 게시되는 AWS 리전입니다.
max-api-requests-per-second	아니요	`1000`	제한이 발생하기 전에 브로커 API가 1초마다 처리할 수 있는 최대 요청 수를 지정합니다.
`enable-throttling-forwarded-for-header`	아니요	`false`	`true`로 설정하면 제한이 X-Forwared-For 헤더(있는 경우)에서 호출자 IP를 검색합니다.
`create-sessions-number-of-retries-on-failure`	아니요	2	NICE DCV 서버 호스트에서 세션 생성 요청이 실패한 후 수행할 최대 재시도 횟수를 지정합니다. 실패 시 재시도를 수행하지 않으려면 0으로 설정합니다.

파라미터 이름	필수	기본값	설명
autorunfile-arguments-max-size	아니요	50	자동 실행 파일에 전달할 수 있는 최대 인수 수를 지정합니다.
autorunfile-arguments-max-argumentlength	아니요	150	각 자동 실행 파일 인수의 최대 길이를 문자 단위로 지정합니다.
enable-persister	예	false	true로 설정하면 브로커 상태 데이터가 외부 데이터베이스에 유지됩니다.
persistence-db	아니요	이는 enable-persistence 가 true로 설정된 경우에만 필수입니다.	유지를 위해 사용할 데이터베이스를 지정합니다. dynamodb 및 mysql 값만 지원됩니다.
dynamodb-region	아니요	enable-persistence 가 true로, persistence-db 가 dynamodb로 설정된 경우에만 필요합니다.	DynamoDB 테이블이 생성되고 액세스되는 리전을 지정합니다.

파라미터 이름	필수	기본값	설명
dynamoc table-rcu	아니요	enable-persistence 가 true로, persisten ce-db 가 dynamodb로 설정된 경우에만 필요합니다.	각 DynamoDB 테이블의 읽기 용량 단위(RCU)를 지정합니다. RCU에 대한 자세한 내용은 프로비저닝된 용량 요금을 참조하세요.
dynamoc table-wcu	아니요	enable-persistence 가 true로, persisten ce-db 가 dynamodb로 설정된 경우에만 필요합니다.	각 DynamoDB 테이블의 쓰기 용량 단위(WCU)를 지정합니다. WCU에 대한 자세한 내용은 프로비저닝된 용량 요금을 참조하세요.
dynamoc table-nam e-prefix	아니요	enable-persistence 가 true로, persisten ce-db 가 dynamodb로 설정된 경우에만 필요합니다.	각 DynamoDB 테이블에 추가되는 접두사를 지정합니다(동일한 AWS 계정을 사용하는 여러 브로커 클러스터를 구분하는 데 유용함). 영숫자와 마침표, 대시, 밑줄 기호만 허용됩니다.

파라미터 이름	필수	기본값	설명
jdbc-connection-url	아니요	enable-persistence 가 true로, persistence-db 가 mysql로 설정된 경우에만 필요합니다.	MariaDB/MySQL 데이터베이스의 연결 URL을 지정합니다. 여기에는 엔드포인트와 데이터베이스 이름이 포함됩니다. URL 형식은 다음과 같아야 합니다. `jdbc:mysql://<db_endpoint>:<db_port>/<db_name>?createDatabaseIfNotExist=true` 여기서 <db_endpoint> 는 MariaDB/MySQL 데이터베이스 엔드포인트, <db_port> 는 데이터베이스 포트, <db_name> 은 데이터베이스 이름입니다.
jdbc-user	아니요	enable-persistence 가 true로, persistence-db 가 mysql로 설정된 경우에만 필요합니다.	MariaDB/MySQL 데이터베이스에 액세스할 수 있는 사용자의 이름을 지정합니다.
jdbc-password	아니요	enable-persistence 가 true로, persistence-db 가 mysql로 설정된 경우에만 필요합니다.	MariaDB/MySQL 데이터베이스에 액세스할 수 있는 사용자의 암호를 지정합니다.

파라미터 이름	필수	기본값	설명
seconds before-deleting-unreachable-dcv-server	아니요	1800	접속할 수 없는 서버가 시스템에서 삭제되는 데 걸리는 시간(초)을 지정합니다.

에이전트 구성 파일

에이전트 구성 파일(Linux용 /etc/dcv-session-manager-agent/agent.conf, Windows용 C:\Program Files\NICE\DCVSessionManagerAgent\conf\agent.conf)에는 세션 관리자 기능을 사용자 지정하도록 구성할 수 있는 파라미터가 포함되어 있습니다. 원하는 텍스트 편집기를 사용하여 구성 파일을 편집할 수 있습니다.

다음 표에는 에이전트 구성 파일의 파라미터가 나열되어 있습니다.

파라미터 이름	필수	기본값	설명
agent.broker_host	예		브로커 호스트의 DNS 이름을 지정합니다.
agent.broker_port	예	8445	브로커와 통신할 포트를 지정합니다.
agent.cafile	아니요		tls_strict 가 true로 설정된 경우에만 필요합니다. TLS 인증서를 검증하는

파라미터 이름	필수	기본값	설명
			데 필요한 인증서(.pem) 파일의 경로를 지정합니다. 브로커에서 에이전트로 자체 서명된 인증서를 복사합니다.
agent.i t_folde	아니요	• /var/lib/dcv-session-manager-agent/init (Linux)	NICE DCV 서버 세션을 만들 때 초기화할 수 있는 사용자 지정 스크립트를 저장하는 데 사용되는 호스트 서버의 폴더 경로를 지정합니다. 절대 경로를 지정해야 합니다. CreateSessions API의 InitFile 요청 파라미터를 사용하는 사용자가 폴더에 접근할 수 있고 파일을 실행할 수 있어야 합니다.
agent.t _strict	아니요	true	엄격한 TLS 검증을 사용해야 하는지 여부를 나타냅니다.
agent.s tware_s tement_ th	아니요		기본 소프트웨어 명령문을 사용하지 않는 경우에만 필요합니다. 소프트웨어 명령문 파일의 경로를 지정합니다. 자세한 내용은 generate-software-statement 섹션을 참조하세요.

파라미터 이름	필수	기본값	설명
agent.t s_folde	아니요	• /etc/dcv-session-manager-agent (Linux) • C:\Program Files\NICE\DCVSessionManagerAgent\conf\tags(Windows)	태그 파일이 있는 폴더의 경로를 지정합니다. 자세한 내용은 태그를 사용하여 NICE DCV 서버를 대상으로 지정 섹션을 참조하세요.
agent.a orun_fo er	아니요	• /var/lib/dcv-session-manager-agent/autorun (Linux) • C:\ProgramData\NICE\DcvSessionManagerAgent\autorun (Windows)	세션 시작 시 자동으로 실행되도록 허용된 스크립트와 앱을 저장하는 데 사용되는 호스트 서버의 폴더 경로를 지정합니다. 절대 경로를 지정해야 합니다. CreateSessions API의 AutorunFile 요청 파라미터를 사용하는 사용자가 폴더에 접근할 수 있고 파일을 실행할 수 있어야 합니다.
agent.m _virtua sessior	아니요	-1 (제한 없음)	NICE DCV 세션 관리자를 사용하여 NICE DCV 서버에 생성할 수 있는 최대 가상 세션 수입니다.
agent.m _concur nt_sess ns_per_ er	아니요	1	NICE DCV 세션 관리자를 사용하여 단일 사용자가 NICE DCV 서버에 생성할 수 있는 최대 가상 세션 수입니다.

파라미터 이름	필수	기본값	설명
agent.broker_update_interval	아니요	30	업데이트된 데이터를 브로커에 전송하기 전에 대기할 시간(초)을 지정합니다. 전송된 데이터에는 업데이트된 세션 정보뿐만 아니라 NICE DCV 서버 및 호스트 상태가 포함됩니다. 값이 낮을수록 세션 관리자는 에이전트가 실행되는 시스템에서 발생하는 변경 사항을 더 잘 인식하지만, 시스템 부하와 네트워크 트래픽이 증가합니다. 값이 높을수록 시스템 및 네트워크 부하가 감소하지만, 시스템 변경에 대한 세션 관리자의 응답성이 떨어지므로 120보다 큰 값은 사용하지 않는 것이 좋습니다.

파라미터 이름	필수	기본값	설명
log.lev	아니요	info	로그 파일 세부 수준을 지정합니다. 다음 세부 수준을 사용할 수 있습니다. • error - 최소 세부 정보를 제공합니다. 오류만 포함합니다. • warning - 오류와 경고를 포함합니다. • info - 기본 세부 수준입니다. 오류, 경고 및 정보 메시지를 포함합니다. • debug - 최대 세부 정보를 제공합니다. 문제를 디버깅하는 데 유용한 세부 정보를 제공합니다.
log.dir tory	아니요	• /var/log/dcv-session-manager-agent/(Linux) • C:\ProgramData\NICE\DCVSessionManagerAgent\log (Windows)	로그 파일을 생성할 디렉터리를 지정합니다.

파라미터 이름	필수	기본값	설명
log.rot ion	아니요	daily	로그 파일 순환을 지정합니다. 유효한 값은 다음과 같습니다. • hourly - 로그 파일이 매시간 순환됩니다. • daily - 로그 파일이 매일 순환됩니다.
log.ma f ile- size	아니요	10485760	로그 파일 크기가 지정된 크기(바이트 단위)에 도달하면 파일이 순환됩니다. 새 로그 파일이 생성되고 새 파일에 추가 로그 이벤트가 배치됩니다.
log.rot e	아니요	9	순환에서 보존되는 최대 로그 파일 수입니다. 순환이 발생하고 이 수에 도달할 때마다 가장 오래된 로그 파일이 삭제됩니다.

NICE DCV 세션 관리자의 릴리스 정보 및 문서 기록

이 페이지에서는 NICE DCV 세션 관리자에 대한 릴리스 정보와 문서 기록을 제공합니다.

주제

NICE DCV 세션 관리자 릴리스 정보

이 섹션에서는 NICE DCV 세션 관리자의 주요 업데이트, 기능 릴리스 정보 및 버그 수정에 대한 개요를 제공합니다. 모든 업데이트는 릴리스 날짜별로 정리되어 있습니다. 사용자로부터 받은 의견을 수렴하기 위해 설명서가 자주 업데이트됩니다.

주제

2023.1— 2023년 11월 9일

빌드 번호	변경 및 버그 수정
• 브로커: 410 • 에이전트: 732 • CLI: 140	• 버그 수정 및 성능 향상

2023.0-15065— 2023년 5월 4일

빌드 번호	변경 및 버그 수정
• 브로커: 392 • 에이전트: 675 • CLI: 132	• ARM 플랫폼에 대한 Red Hat Enterprise Linux 9, Rocky Linux 9, CentOS Stream 9 지원이 추가되었습니다.

2023.0-14852— 2023년 3월 28일

빌드 번호	변경 및 버그 수정
• 브로커: 392 • 에이전트: 642 • CLI: 132	• Red Hat Enterprise Linux 9, Rocky Linux 9, CentOS Stream 9 지원이 추가되었습니다.

2022.2-13907 — 2022년 11월 11일

빌드 번호	변경 및 버그 수정
• 브로커: 382 • 에이전트: 612 • CLI: 123	• `DescribeSessions` 응답에 `Substate` 필드가 추가되었습니다. • 사용 중인 URL에 따라 CLI가 브로커에 연결되지 않는 문제를 해결했습니다.

2022.1-13067— 2022년 6월 29일

빌드 번호	변경 및 버그 수정
• 브로커: 355 • 에이전트: 592 • CLI: 114	• AWS Graviton 인스턴스에서 브로커를 실행하기 위한 지원이 추가되었습니다. • Ubuntu 22.04에 에이전트 및 브로커 지원이 추가되었습니다.

2022.0-11952 — 2022년 2월 23일

빌드 번호	변경 및 버그 수정
• 브로커: 341 • 에이전트: 520 • CLI: 112	• 에이전트에 로그 순환 기능을 추가했습니다. • 브로커에서 Java 홈을 설정하는 구성 파라미터를 추가했습니다. • 브로커의 캐시에서 디스크로의 데이터 플러시를 개선했습니다. • CLI에서 URL 검증을 수정했습니다.

2021.3-11591— 2021년 12월 20일

빌드 번호	새로운 기능
• 브로커: 307 • 에이전트: 453 • CLI: 92	• NICE DCV 연결 게이트웨이 통합에 대한 지원이 추가되었습니다. • Ubuntu 18.04 및 Ubuntu 20.04에 대한 브로커 지원이 추가되었습니다.

2021.2-11445— 2021년 11월 18일

빌드 번호	변경 및 버그 수정
• 브로커: 288 • 에이전트: 413 • CLI: 54	• Windows 도메인이 포함된 로그인 이름을 검증할 때 발생하는 문제를 해결했습니다.

2021.2-11190— 2021년 10월 11일

빌드 번호	변경 및 버그 수정
• 브로커: 254 • 에이전트: 413 • CLI: 54	• Windows 세션을 시작하지 못하게 하는 명령줄 인터페이스 문제를 해결했습니다.

2021.2-11042— 2021년 9월 1일

빌드 번호	새로운 기능	변경 및 버그 수정
• 브로커: 254 • 에이전트: 413 • CLI: 37	• 이제 NICE DCV 세션 관리자가 명령줄 인터페이스(CLI) 지원을 제공합니다. API를 호출하는 대신 CLI에서 NICE DCV 세션을 생성하고 관리할 수 있습니다. • NICE DCV 세션 관리자에 브로커 데이터 지속성이 도입되었습니다. 가용성을 높이기 위해 브로커는 외부 데이터 저장소에 서버 상태 정보를 유지하고 시작 시 데이터를 복원할 수 있습니다.	• 이제 외부 권한 부여 서버를 등록할 때 권한 부여 서버가 JSON 형식의 웹 토큰에 서명하는 데 사용하는 알고리즘을 지정할 수 있습니다. 이번 변경으로 Azure AD를 외부 권한 부여 서버로 사용할 수 있습니다.

2021.1-10557— 2021년 5월 31일

빌드 번호	새로운 기능	변경 및 버그 수정
• 브로커: 214 • 에이전트: 365	• NICE DCV 세션 관리자에 Linux에서 자동 실행 파일에 전달되는 입력 파라미터에 대한 지원이 추가되었습니다. • 이제 서버 속성을 CreateSessions API에 요구 사항으로 전달할 수 있습니다.	• Windows의 자동 실행 파일 관련 문제를 해결했습니다.

2021.0-10242— 2021년 4월 12일

빌드 번호	변경 및 버그 수정

• 브로커: 183
• 에이전트: 318
	• NICE DCV 세션 관리자에 다음과 같은 새로운 API가 도입되었습니다.
	  • OpenServers
	  • CloseServers
	  • DescribeServers
	  • GetSessionScreenshots
	• 또한 다음과 같은 새 구성 파라미터가 도입되었습니다.
	  • 브로커 파라미터: `session-screenshot-max-width` , `session-screenshot-max-height` , `session-screenshot-format` , `create-sessions-queue-max-size` , `create-sessions-queue-max-time-seconds`
	  • 에이전트 파라미터: `agent.autorun_folder` , `max_virtual_sessions` , `max_concurrent_sessions_per_user`
	
	  에이전트 파라미터: `agent.autorun_folder` ,`max_virtual_sessions` , `max_concurrent_sessions_per_user`
	
	  에이전트 파라미터: `agent.autorun_folder` ,`max_virtual_sessions` , `max_concurrent_sessions_per_user`

2020.2-9662— 2020년 12월 4일

빌드 번호	변경 및 버그 수정
• 브로커: 114 • 에이전트: 211	• 브로커가 시작되지 못하게 하는 자동 생성 TLS 인증서 문제를 해결했습니다.

2020.2-9508— 2020년 11월 11일

빌드 번호	변경 및 버그 수정
• 브로커: 78 • 에이전트: 183	• NICE DCV 세션 관리자의 첫 릴리스입니다.

문서 기록

다음 표는 NICE DCV 세션 관리자의 본 릴리스 관련 설명서를 소개합니다.

변경 사항	설명	날짜
NICE DCV 버전 2023.1	NICE DCV 세션 관리자가 NICE DCV 2023.1에 맞게 업데이트되었습니다. 자세한 내용은 2023.1— 2023년 11월 9일 섹션을 참조하세요.	2023년 11월 9일
NICE DCV 버전 2023.0	NICE DCV 세션 관리자가 NICE DCV 2023.0에 맞게 업데이트되었습니다. 자세한 내용은 2023.0-14852— 2023년 3월 28일 섹션을 참조하세요.	2023년 3월 28일
NICE DCV 버전 2022.2	NICE DCV 세션 관리자가 NICE DCV 2022.2에 맞게 업데이트되었습니다. 자세한 내용은 2022.2-13907 — 2022년 11월 11일 섹션을 참조하세요.	2022년 11월 11일
NICE DCV 버전 2022.1	NICE DCV 세션 관리자가 NICE DCV 2022.1에 맞게 업데이트되었습니다. 자세한 내용은 2022.1-13067— 2022년 6월 29일 섹션을 참조하세요.	2022년 6월 29일
NICE DCV 버전 2022.0	NICE DCV 세션 관리자가 NICE DCV 2022.0에 맞게 업데이트되었습니다. 자	2022년 2월 23일

변경 사항	설명	날짜
	세한 내용은 2022.0-11952 — 2022년 2월 23일 섹션을 참조하세요.	
NICE DCV 버전 2021.3	NICE DCV 세션 관리자가 NICE DCV 2021.3에 맞게 업데이트되었습니다. 자세한 내용은 2021.3-11591— 2021년 12월 20일 섹션을 참조하세요.	2021년 12월 20일
NICE DCV 버전 2021.2	NICE DCV 세션 관리자가 NICE DCV 2021.2에 맞게 업데이트되었습니다. 자세한 내용은 2021.2-11042— 2021년 9월 1일 섹션을 참조하세요.	2021년 9월 1일
NICE DCV 버전 2021.1	NICE DCV 세션 관리자가 NICE DCV 2021.1에 맞게 업데이트되었습니다. 자세한 내용은 2021.1-10557— 2021년 5월 31일 섹션을 참조하세요.	2021년 5월 31일
NICE DCV 버전 2021.0	NICE DCV 세션 관리자가 NICE DCV 2021.0에 맞게 업데이트되었습니다. 자세한 내용은 2021.0-10242— 2021년 4월 12일 섹션을 참조하세요.	2021년 4월 12일
NICE DCV 세션 관리자의 첫 릴리스	이 내용의 첫 번째 발행입니다.	2020년 11월 11일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.