Tableau Server 엔터프라이즈 배포 가이드

마지막 업데이트 2024-07-25

콘텐츠

Tableau Server 엔터프라이즈 배포 가이드

Tableau Server EDG(엔터프라이즈 배포 가이드)는 Tableau Server 배포(온프레미스 또는 클라우드)를 위한 규범적 지침을 제공하기 위해 개발되었습니다. 이 가이드는 참조 아키텍처의 컨텍스트에서 엔터프라이즈 시나리오에 대한 배포 지침을 제공합니다. Tableau는 참조 아키텍처를 테스트하여 업계 표준 모범 사례에 따른 보안, 확장성 및 성능 벤치마크를 준수하는지 확인했습니다.

개괄적으로 업계 표준 엔터프라이즈 배포의 핵심 기능은 서버 응용 프로그램 기능의 각 계층(웹 게이트웨이 계층, 응용 프로그램 계층 및 데이터 계층)이 액세스 제어 서브넷으로 바인딩되고 보호되는 계층형 토폴로지로 구성됩니다. 인터넷에서 서버 응용 프로그램에 액세스하는 사용자는 웹 계층에서 인증됩니다. 인증되면 보호되는 서브넷으로 요청이 중계되고 여기에서 응용 프로그램 계층이 비즈니스 논리를 처리합니다. 가치가 높은 데이터는 세 번째 서브넷인 데이터 계층으로 보호됩니다. 응용 프로그램 계층의 서비스는 보호되는 네트워크를 통해 데이터 계층과 통신하여 백엔드 데이터 원본에 대한 데이터 요청을 처리합니다.

이 배포에서는 모든 설계 의사 결정 및 구현에서 보안을 가장 먼저 고려합니다. 그러나 안정성, 성능 및 확장성도 우선적인 요구 사항입니다. 이 참조 아키텍처의 분산형 및 모듈식 설계를 보면 안정성 및 성능이 예측 가능한 방식으로 선형 확장되는 것을 알 수 있습니다. 호환되는 서비스를 각 노드에 전략적으로 함께 배치하고 요충지에 서비스를 추가하기 때문입니다.

이 가이드의 대상

EDG는 다음을 수행해야 하는 엔터프라이즈 IT 관리자를 위해 개발되었습니다.

- IT에서 관리하는 Tableau 배포
- 업계 규정 준수 정책 적용

- 업계 배포 모범 사례
- 기본적으로 안전한 배포

EDG는 엔터프라이즈 참조 아키텍처 배포를 위한 구축 가이드입니다. 이 EDG 버전에는 AWS/Linux 구축 예가 포함되어 있지만, 이 가이드는 숙련된 엔터프라이즈 IT 관리자가 특정 참조 아키텍처를 모든 업계 표준 데이터 센터 환경에 배포하는 리소스로 사용할 수 있습니다.

버전

이 버전의 EDG는 Tableau Server의 2021.2.3 버전 이상을 위해 개발되었습니다. 이전 버전의 Tableau Server를 배포할 때 이 EDG를 일반 참조로 사용할 수 있지만 2021.2.3 이상이 포함된 참조 아키텍처를 배포하는 것이 좋습니다. 이전 버전의 Tableau Server에서는 일부 기능 및 옵션을 사용할 수 없습니다.

최신 기능과 개선 사항을 활용하려면 Tableau Server 2022.1.7 이상이 포함된 EDG를 배포하는 것이 좋습니다.

이 가이드에 설명된 참조 아키텍처는 다음 Tableau 클라이언트(호환되는 브라우저가 포함된 웹 작성, Tableau Mobile 및 Tableau Desktop 버전 2021.2.1 이상)를 지원합니다. 다른 Tableau 클라이언트(Tableau Prep, Bridge 등)는 아직 참조 아키텍처를 통해 검증되지 않았습니다.

주요 기능

Tableau Server 참조 아키텍처의 첫 번째 버전은 다음 시나리오 및 기능을 도입합니다.

- 클라이언트 사전 인증: Tableau 클라이언트(Desktop, Mobile, 웹 작성)는 내부 Tableau Server에 액세스하기 전에 웹 계층에서 기업 인증 공급자를 통해 인증됩니다. 이 프로세스는 역방향 프록시 서버 역할을 하는 Tableau Server 독립 게이트웨이에서 authN 플러그인을 구성하여 관리됩니다. 5부 - 웹 계층 구성을 참조하십시오.
- 제로 트러스트 배포: Tableau Server에 대한 모든 트래픽이 사전 인증되기 때문에 전체 Tableau 배포가 신뢰할 수 있는 연결이 필요하지 않은 사설 서브넷에서 작동합니다.

- 외부 리포지토리: 참조 아키텍처는 Tableau 리포지토리를 외부 PostgreSQL 데이터베이스에 설치하도록 지정하여 DBA가 리포지토리를 일반 데이터베이스로 관리, 최적화, 확장 및 백업할 수 있도록 합니다.
- 초기 노드 복구: EDG는 장애 발생 시 초기 노드 복원을 자동화하는 스크립트를 도입합니다.
- tar 기반 백업 및 복원: 익숙한 tar 백업을 Tableau 배포의 전략적 이정표로 사용합니다. 실패 또는 배포 구성 오류가 발생하는 경우 연결된 tar 백업을 복구하여 이전 배포 단계로 빠르게 복구할 수 있습니다.
- 성능 개선: 고객과 연구실 검증에서 EDG를 실행할 때 표준 배포에 비해 15%~20% 성능이 향상되는 것으로 나타났습니다.

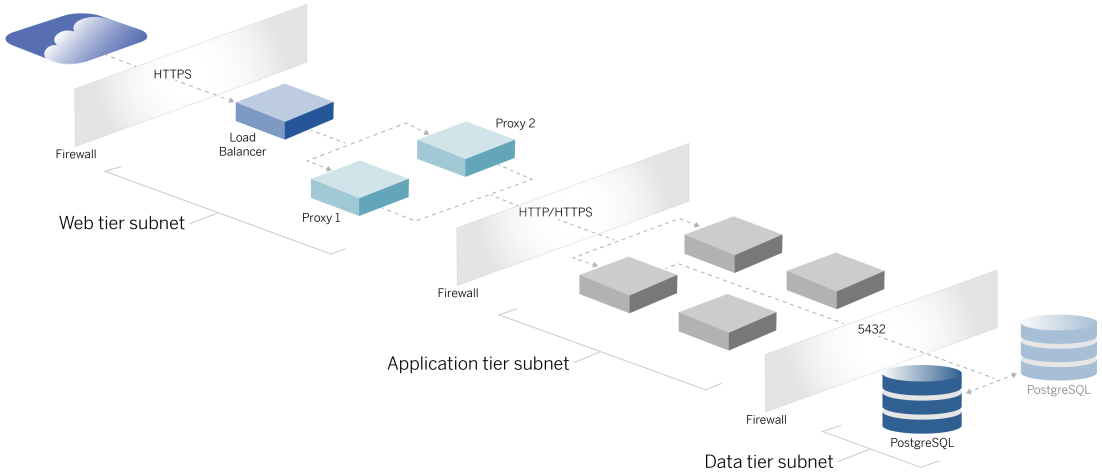
라이선스

이 가이드에 규정된 Tableau Sever 참조 아키텍처를 사용하려면 Tableau Advanced Management 라이선스를 통해 Tableau Server 외부 리포지토리를 사용할 수 있어야 합니다. 필요한 경우 Tableau Server 외부 파일 저장소를 배포할 수 있는데, 이 경우에도 Tableau Advanced Management 라이선스가 필요합니다. *Tableau Server(Linux)의 Tableau Advanced Management 정보*를 참조하십시오.

1부 - 엔터프라이즈 배포 이해

1부에서는 Tableau Server 엔터프라이즈 배포 가이드가 설계된 이유인 업계 표준 엔터프라이즈 배포의 기능과 요구 사항을 보다 자세히 설명합니다.

다음 네트워크 다이어그램은 Tableau Server 참조 아키텍처를 사용한 일반적인 데이터 센터 계층형 배포를 보여줍니다.



업계 표준 및 배포 요구 사항

다음은 업계 표준 배포의 기능입니다. 참조 아키텍처는 다음과 같은 요구 사항을 충족하도록 설계되었습니다.

- 다중 계층 네트워크 설계: 네트워크는 웹 계층, 응용 프로그램 계층 및 데이터 계층의 각 계층에서 액세스를 제한하기 위해 보호되는 서브넷으로 바인딩됩니다. 모든 통신은 다음 서브넷에서 종료되므로 어떤 통신도 서브넷을 통과할 수 없습니다.
- 포트 및 프로토콜을 기본적으로 차단: 각 서브넷 또는 보안 그룹은 모든 인바운드 및 아웃바운드 포트와 프로토콜을 기본적으로 차단합니다. 통신을 사용하려면 부분적으로 포트 또는 프로토콜 구성에서 예외를 열어야 합니다.
- 개별 웹 인증: 인터넷의 사용자 요청은 웹 계층의 역방향 프록시에 있는 인증 모듈에 의해 인증됩니다. 따라서 응용 프로그램 계층에 대한 모든 요청은 웹 계층에서 인증된 후 보호되는 응용 프로그램 계층으로 전달됩니다.

- 플랫폼 독립성: 온프레미스 서버 응용 프로그램 또는 클라우드에 솔루션을 배포할 수 있습니다.
- 기술 불가지론: 가상 컴퓨터 환경 또는 컨테이너에 솔루션을 배포할 수 있습니다. Windows 또는 Linux에도 배포할 수 있습니다. 그러나 이 초기 버전의 참조 아키텍처와 지원 문서는 AWS에서 실행되는 Linux용으로 개발되었습니다.
- 고가용성: 시스템의 모든 구성 요소는 클러스터로 배포되며 활성/활성 또는 활성/비활성 배포에서 작동하도록 설계되었습니다.
- 역할 분리: 각 서버는 고유한 역할을 수행합니다. 이 설계는 모든 서버를 분할하여 서비스 관련 관리자로 액세스를 최소화할 수도 있습니다. 예를 들어 DBA는 Tableau용 PostgreSQL을 관리하고, ID 관리자는 웹 계층의 인증 모듈을 관리하며, 네트워크 및 클라우드 관리자는 트래픽과 연결을 사용하도록 설정합니다.
- 선형 확장: 고유한 역할과 마찬가지로 각 계층 서비스를 로드 프로필에 따라 독립적으로 확장할 수 있습니다.
- 클라이언트 지원: 참조 아키텍처는 모든 Tableau 클라이언트(Tableau Desktop 버전 2021.2 이상, Tableau Mobile 및 Tableau 웹 작성)를 지원합니다.

보안 수단

언급한 바와 같이 업계 표준 데이터 센터 설계의 주된 기능은 보안입니다.

- 액세스: 각 계층은 네트워크 계층에서 포트 필터링을 사용하여 액세스 제어를 적용하는 서브넷으로 바인딩됩니다. 응용 프로그램 계층에서는 프로세스 간 서비스 인증을 통해 서브넷 간 통신 액세스에도 인증을 적용할 수 있습니다.
- 통합: 아키텍처는 웹 계층의 역방향 프록시에서 ID 공급자(IdP)로 플러그인하도록 설계되었습니다.
- 개인 정보 보호: 웹 계층으로의 트래픽은 클라이언트에서 SSL을 통해 암호화됩니다. 필요한 경우 내부 서브넷으로의 트래픽도 암호화할 수 있습니다.

웹 프록시 계층

웹 계층은 DMZ(매개 변수 영역이라고도 함)의 서브넷으로, 인터넷과 응용 프로그램이 배포된 내부 서브넷 간의 보안 완충 지대 역할을 합니다. 웹 계층은 민감한 정보를 저장하지 않는 역방향 프록시 서버를 호스팅합니다. 역방향 프록시 서버는 AuthN 플러그인으로 구성되어 클라이언트 요청을 Tableau Server로 리디렉션하기 전에 신뢰할 수 있는

IdP를 사용하여 클라이언트 세션을 사전 인증합니다. 자세한 내용은 AuthN 모듈을 사용한 사전 인증을 참조하십시오.

부하 분산 장치

이 배포 설계에서는 엔터프라이즈 부하 분산 솔루션을 역방향 프록시 서버 앞에 포함합니다.

부하 분산 장치는 다음을 통해 중요한 보안 및 성능 개선을 제공합니다.

- 응용 프로그램 계층 서비스의 프런트엔드 URL 가상화
- SSL 암호화 적용
- SSL 오프로드
- 클라이언트와 웹 계층 서비스 간에 압축 적용
- DOS 공격으로부터 보호
- 고가용성 제공

참고: Tableau Server 버전 2022.1에는 Tableau Server 독립 게이트웨이가 포함됩니다. 독립 게이트웨이는 Tableau 게이트웨이 프로세스의 독립 실행형 인스턴스로, Tableau 인식 역방향 프록시 역할을 합니다. 릴리스 시점에 독립 게이트웨이는 검증되었지만 EDG 참조 아키텍처에서 완벽하게 테스트되지 않았습니다. 전체 테스트가 완료되면 EDG가 Tableau Server 독립 게이트웨이 규범적 지침으로 업데이트될 것입니다.

응용 프로그램 계층

응용 프로그램 계층은 서버 응용 프로그램의 핵심 비즈니스 논리를 실행하는 서브넷입니다. 응용 프로그램 계층은 클러스터의 분산 노드 전체에 걸쳐 구성되는 서비스 및 프로세스로 구성됩니다. 응용 프로그램은 웹 계층에서만 액세스할 수 있고 사용자는 직접 액세스할 수 없습니다.

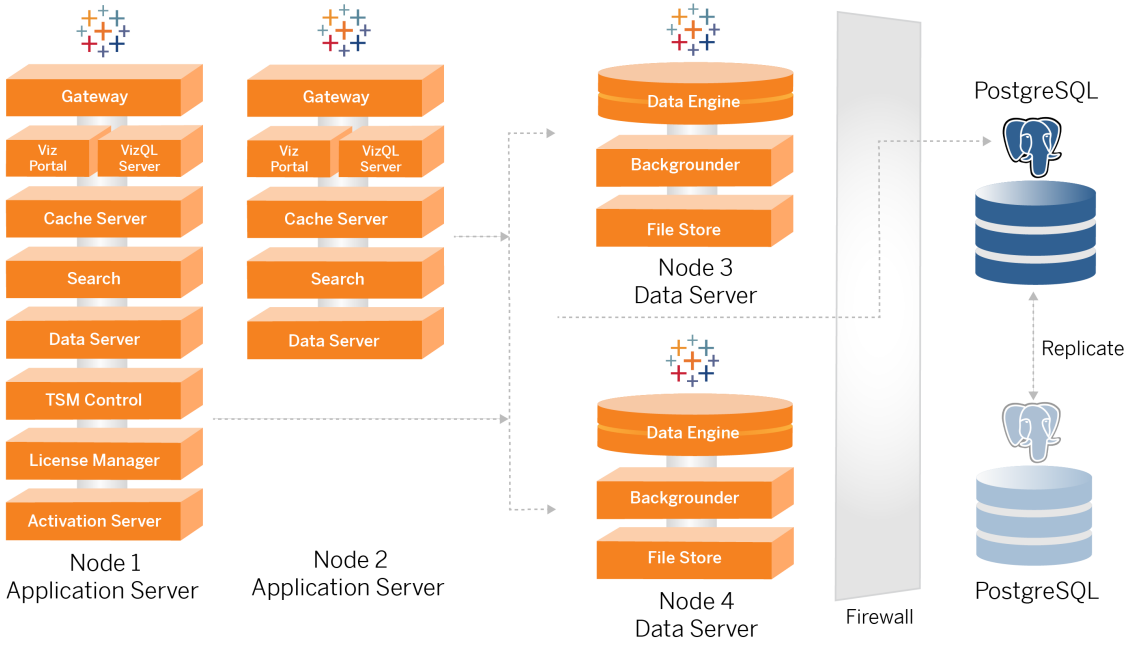
리소스 사용 프로필이 다른 프로세스(예: CPU 집약적 프로세스와 메모리 집약적 프로세스)를 함께 배치하도록 응용 프로그램 프로세스를 구성하면 성능 및 안정성이 개선됩니다.

데이터 계층

데이터 계층은 중요한 데이터를 보관하는 계층입니다. 이 계층으로의 모든 트래픽은 응용 프로그램 계층에서 시작되므로 이미 인증이 완료된 상태입니다. 네트워크 계층의 액세스 요구 사항 및 포트 구성뿐 아니라 이 계층에는 인증된 액세스 권한이 포함되어야 하고 선택적으로 응용 프로그램 계층과 함께 암호화된 트래픽이 포함되어야 합니다.

2부 - Tableau Server 배포 참조 아키텍처 이해

다음 이미지에서는 관련 Tableau Server 프로세스와 참조 아키텍처에 이러한 프로세스가 배포되는 방식을 보여 줍니다. 이 배포는 엔터프라이즈에 적합한 최소 Tableau Server 배포로 간주됩니다.



이 항목의 프로세스 다이어그램은 각 노드의 주요 정의 프로세스를 보여 주기 위한 것입니다. 다이어그램에 표시되지 않은 노드에서도 실행되는 지원 프로세스가 많이 있습니다. 모든 프로세스 목록은 이 가이드의 구성 섹션에서 4부 - Tableau Server 설치 및 구성을 참조하십시오.

Tableau Server 프로세스

Tableau Server 참조 아키텍처는 PostgreSQL에 외부 리포지토리가 있는 4노드 Tableau Server 클러스터 배포입니다.

- Tableau Server 초기 노드(노드 1): 클러스터의 단일 노드에서만 실행할 수 있는 TSM 관리 및 라이선스 서비스가 필요합니다. 엔터프라이즈 환경에서 Tableau Server 초기 노드는 클러스터의 주 노드입니다. 이 노드는 노드 2에서도 중복 응용 프로그램 서비스를 실행합니다.
- Tableau Server 응용 프로그램 노드(노드 1 및 노드 2): 이 두 노드는 클라이언트 요청을 처리하고 데이터 원본 및 데이터 노드에 연결하여 쿼리를 수행합니다.
- Tableau Server 데이터 노드(노드 3 및 노드 4): 데이터 관리를 전담하는 노드입니다.
- 외부 PostgreSQL: 이 호스트에서는 Tableau Server 리포지토리 프로세스를 실행합니다. HA 배포를 위해서는 활성/비활성 중복성을 위해 PostgreSQL 호스트를 추가로 실행해야 합니다.

  Amazon RDS에서 PostgreSQL을 실행할 수도 있습니다. RDS에서 리포지토리를 실행하는 경우와 EC2 인스턴스에서 리포지토리를 실행하는 경우 간의 차이점에 대한 자세한 내용은 *Tableau Server 외부 리포지토리*(Linux)를 참조하십시오.

  외부 리포지토리와 함께 Tableau Server를 배포하려면 Tableau Advanced Management 라이선스가 필요합니다.

  사내 DBA 전문가가 없는 조직의 경우 기본 내부 PostgreSQL 구성에서 Tableau Server 리포지토리 프로세스를 선택적으로 실행할 수도 있습니다. 기본 시나리오에서 리포지토리는 PostgreSQL이 내장된 Tableau 노드에서 실행됩니다. 이 경우 전용 Tableau 노드에서 리포지토리를 실행하고, 리포지토리 장애 조치를 지원하기 위해 추가 전용 노드에서 비활성 리포지토리를 실행하는 것이 좋습니다. *리포지토리 장애 조치*(Linux)를 참조하십시오.

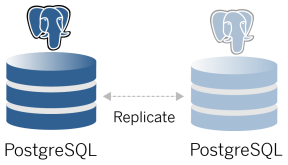
  이 가이드에 예제로 설명된 AWS 구현은 EC2 인스턴스에서 실행되는 PostgreSQL에 외부 리포지토리를 배포하는 방법을 설명합니다.

- 선택 사항: 조직에서 외부 스토리지를 사용하는 경우 Tableau 파일 저장소를 외부 서비스로 배포할 수 있습니다. 이 가이드에서는 핵심 배포 시나리오에 외부 파일 저장소를 포함하지 않습니다. *Tableau Server에 외부 파일 저장소 설치*(Linux)를 참조하십시오.

외부 파일 저장소와 함께 Tableau Server를 배포하려면 Tableau Advanced Management 라이선스가 필요합니다.

PostgresSQL 리포지토리

Tableau Server 리포지토리는 서버 데이터를 저장하는 PostgresSQL 데이터베이스입니다. 이 데이터에는 Tableau Server 사용자, 그룹 및 그룹 할당, 사용 권한, 프로젝트, 데이터 원본, 추출 메타데이터 및 새로 고침 정보와 관련된 정보가 포함됩니다.



기본 PostgresSQL 배포에는 시스템 메모리 리소스의 거의 50%가 사용됩니다. 사용 유형(프로덕션 및 대규모 프로덕션 배포)에 따라 리소스 사용량이 증가할 수 있습니다. 이러한 이유로 VizQL, 백그라운더 또는 데이터 엔진과 같은 다른 리소스 집약적 서버 구성 요소를 실행하지 않는 컴퓨터에서 리포지토리 프로세스를 실행하는 것이 좋습니다. 이러한 구성 요소와 함께 리포지토리 프로세스를 실행하면 IO 경합이 발생하여 리소스가 제약되고 배포의 전체 성능이 저하됩니다.

노드 1: 초기 노드

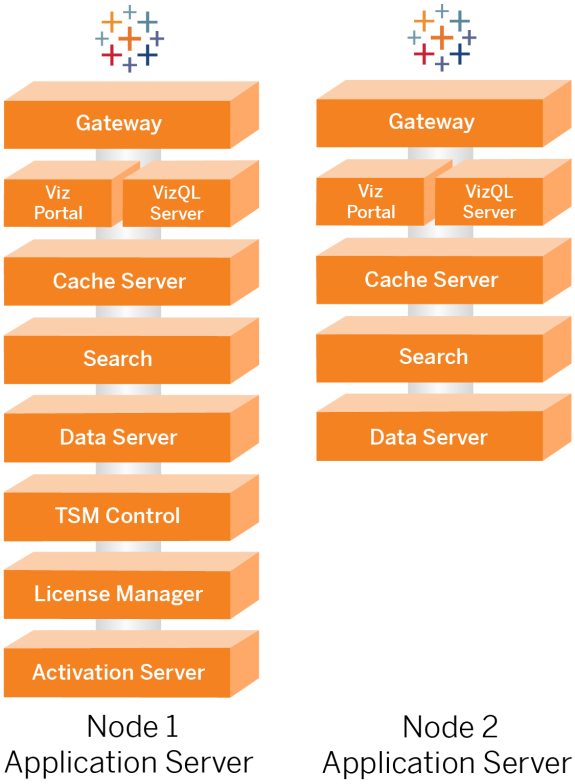
초기 노드는 소수의 중요한 프로세스를 실행하고 응용 프로그램 노드를 노드 2와 공유합니다.

Tableau를 설치한 첫 번째 컴퓨터인 "초기 노드"에는 몇 가지 고유한 특성이 있습니다. 라이선스 서비스(라이선스 관리자), 활성화 서비스 및 TSM 컨트롤러(관리 컨트롤러)의 세 프로세스는 초기 노드에서만 실행되며 장애 상황 이외에는 다른 노드로 이동할 수 없습니다.

노드 1 장애 조치 및 자동 복원

라이선스, 활성화 및 TSM 컨트롤러 서비스는 Tableau Server 배포의 상태에 매우 중요합니다. 노드 1에 장애가 발생해도 사용자가 Tableau Server 배포에 연결할 수 있습니다. 올바르게 구성된 참조 아키텍처가 요청을 노드 2로 라우팅하기 때문입니다. 그러나 이러한 코어 서비스가 없으면 배포가 심각한 장애 보류 상태에 있게 됩니다. 초기 노드 자동 복구를 참조하십시오.

노드 1 및 2: 응용 프로그램 서버



노드 1 및 2는 클라이언트 요청을 처리하고 데이터 원본을 쿼리하고 비주얼리제이션을 생성하며 콘텐츠 및 관리와 기타 핵심 Tableau 비즈니스 논리를 처리하는 Tableau Server 프로세스를 실행합니다. 응용 프로그램 서버는 사용자 데이터를 저장하지 않습니다.

참고: "응용 프로그램 서버"는 TSM에 나열되는 Tableau Sever 프로세스를 나타내는 용어이기도 합니다. "응용 프로그램 서버"의 기초 프로세스는 VizPortal입니다.

부하 분산 논리 실행의 요청을 역방향 프록시 서버에서 처리하도록 노드 1과 노드 2 병렬 실행을 확장합니다. 중복 노드로서 이러한 노드 중 하나에 장애가 발생하면 나머지 노드에서 클라이언트 요청 및 서비스를 처리합니다.

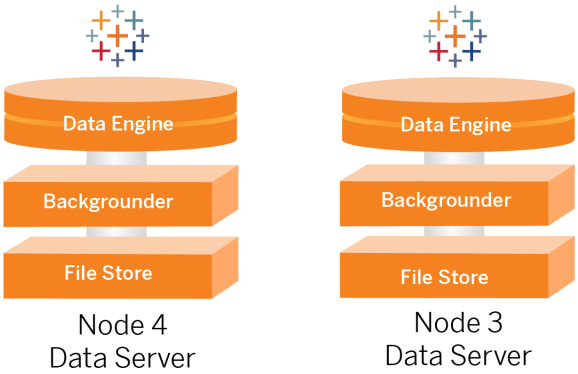
보조 응용 프로그램 프로세스가 동일한 컴퓨터에서 실행되도록 참조 아키텍처가 설계됩니다. 즉, 프로세스가 컴퓨팅 리소스를 놓고 경쟁하지 않고 경합을 유발하지 않는다는 의미입니다.

예를 들어 응용 프로그램 서버의 코어 처리 서비스인 VizQL은 CPU 및 메모리 제한이 높은 프로세스로, 컴퓨터 CPU 및 메모리의 거의 60~70%를 사용합니다. 이러한 이유로 이 참조 아키텍처는 다른 메모리 또는 CPU 제한 프로세스를 VizQL과 동일한 노드에서 실행하지 않도록 설계되었습니다. 테스트에서 부하의 양 또는 사용자 수는 VizQL 노드의 메모리 또는 CPU 사용량에 영향을 미치지 않는 것으로 밝혀졌습니다. 예를 들어 부하 테스트에서 동시 사용자 수를 줄이면 대시보드 또는 비주얼리제이션 로드 프로세스의 성능만 영향을 받고 리소스 사용률이 줄어들지는 않습니다. 따라서 사용량이 피크일 때의 가용 메모리 및 CPU에 따라 더 많은 VizQL 프로세스를 추가하는 것을 고려할 수 있습니다. 일반적인 통합 문서로 시작할 때는 VizQL 프로세스당 4개의 코어를 할당합니다.

응용 프로그램 서버 확장

참조 아키텍처는 사용 기반 모델에 따라 확장할 수 있도록 설계되었습니다. 일반적으로 최소 2개의 응용 프로그램 서버를 사용하는 것이 좋습니다. 각 서버는 최대 1000명의 사용자를 지원합니다. 사용자 기반이 증가하는 경우 1000명의 사용자를 추가할 때마다 응용 프로그램 서버를 추가할 계획을 세우십시오. 사용 현황 및 성능을 모니터링하여 조직의 호스트별 사용자 기반을 조정하십시오.

노드 3 및 4: 데이터 서버



파일 저장소, 데이터 엔진(Hyper) 및 백그라운더 프로세스는 다음과 같은 이유로 노드 3과 4에 함께 배치됩니다.

- 추출 최적화: 백그라운더, Hyper 및 파일 저장소를 동일한 노드에서 실행하면 성능 및 안정성이 최적화됩니다. 추출 프로세스 중에 백그라운더는 대상 데이터베이스를 쿼리하고 동일한 노드에 Hyper 파일을 만든 다음 파일 저장소에 업로드합니다. 이러한 프로세스를 동일한 노드에 함께 배치하면 추출 생성 워크플로우에서 다량의 데이터 스트림을 네트워크 또는 노드에 복사하지 않아도 됩니다.
- 보조 리소스 분산: 백그라운더는 대부분 CPU 집약적입니다. 데이터 엔진은 메모리 집약적인 프로세스입니다. 이러한 프로세스를 결합하면 각 노드의 리소스 활용률을 최대화할 수 있습니다.
- 데이터 프로세스 통합: 이러한 각 프로세스는 백엔드 데이터 프로세스이므로 가장 안전한 데이터 계층에서 실행하는 것이 적절합니다. 향후 버전의 참조 아키텍처에서는 응용 프로그램과 데이터 서버가 별도의 계층에서 실행됩니다. 그러나 Tableau 아키텍처의 응용 프로그램 종속성으로 인해 현재는 응용 프로그램 및 데이터 서버가 같은 계층에서 실행되어야 합니다.

데이터 서버 확장

응용 프로그램 서버와 마찬가지로 Tableau 데이터 서버에 필요한 리소스를 계획하려면 사용 기반 모델링이 필요합니다. 일반적으로 각 데이터 서버가 하루에 최대 2000개의 추출 새로 고침 작업을 지원할 수 있다고 가정합니다. 추출 작업이 늘어나면 파일 저장소 서비스 없이 데이터 서버를 더 추가하십시오. 일반적으로 로컬 파일 시스템을 파일 저장소 서비스로 사용하는 배포의 경우 2노드 데이터 서버 배포가 적합합니다. 응용 프

로그램 서버를 더 추가해도 데이터 서버의 성능이나 선형 확장에는 영향을 미치지 않습니다. 실제로 추가 사용자 쿼리로 인한 오버헤드를 제외하고 응용 프로그램 호스트 및 사용자를 더 추가할 경우 미치는 영향은 매우 적습니다.

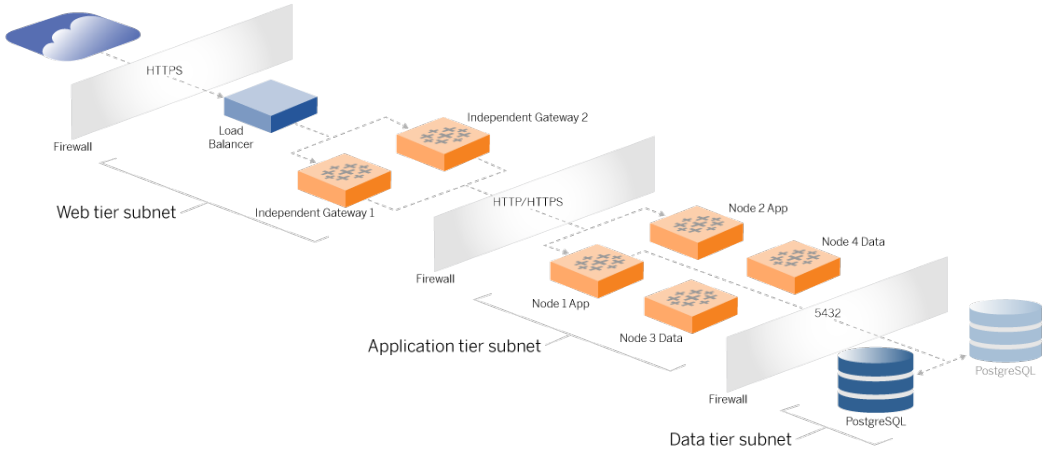
3부 - Tableau Server 엔터프라이즈 배포 준비

3부에서는 Tableau Server 참조 아키텍처를 배포할 인프라를 준비할 때의 요구 사항을 설명합니다. 시작하기 전에 2부 - Tableau Server 배포 참조 아키텍처 이해를 검토하는 것이 좋습니다.

이 항목에서는 요구 사항에 대한 설명 외에도 AWS 환경에서 참조 아키텍처의 구현 예를 제공합니다. 이 가이드의 나머지 부분은 이 항목에서 처음 설명된 AWS 참조 아키텍처 예를 기반으로 합니다.

이 참조 아키텍처의 핵심 원칙은 데이터 센터 보안 모범 사례를 통한 표준화입니다. 이 아키텍처는 서비스를 보호되는 네트워크 서브넷으로 분리하도록 설계되었습니다. 서브넷 간 통신은 특정 프로토콜 및 포트 트래픽으로 제한됩니다.

다음 다이어그램은 온프레미스 배포 또는 고객 관리형 클라우드 배포에 대한 참조 아키텍처 서브넷 설계를 보여줍니다. 클라우드 배포 예제는 아래의 예: AWS에서 서브넷 및 보안 그룹 구성 섹션을 참조하십시오.



서브넷

3개의 서브넷을 만듭니다.

- 웹 계층
- 응용 프로그램 계층
- 데이터 서브넷

방화벽/보안 그룹 규칙

아래 탭은 데이터 센터의 각 계층에 대한 방화벽 규칙을 설명합니다. AWS별 보안 그룹 규칙에 대해서는 이 항목의 뒷부분에 있는 섹션을 참조하십시오.

웹 계층

웹 계층은 인바운드 HTTPS 요청을 처리하고 응용 프로그램 계층으로 요청을 중계하는 공개 DMZ 서브넷입니다. 이 설계는 조직을 표적으로 할 수 있는 악성 프로그램으로부터의 방어 계층을 제공합니다. 웹 계층은 응용 프로그램/데이터 계층에 대한 액세스를 차단합니다.

트래픽	유형	프로토콜	포트 범위	원본
인바운드	SSH	TCP	22	배스천 서브넷(클라우드 배포용)
인바운드	HTTP	TCP	80	인터넷(0.0.0.0/0)
인바운드	HTTPS	TCP	443	인터넷(0.0.0.0/0)
아웃바운드	모든 트래픽	전체	전체	

응용 프로그램 계층

응용 프로그램 서브넷은 Tableau Server 배포가 상주하는 위치입니다. 응용 프로그램 서브넷에는 Tableau 응용 프로그램 서버(노드 1 및 노드 2)가 포함됩니다. Tableau 응용 프로그램 서버는 데이터 서버에 대한 사용자 요청을 처리하고 핵심 비즈니스 논리를 실행합니다.

응용 프로그램 서브넷에는 Tableau 데이터 서버(노드 3 및 노드 4)도 포함됩니다.

응용 프로그램 계층에 대한 모든 클라이언트 트래픽은 웹 계층에서 인증됩니다. 응용 프로그램 서브넷에 대한 관리 액세스는 인증되고 배스천 호스트를 거쳐 라우팅됩니다.

트래픽	유형	프로토콜	포트 범위	원본
인바운드	SSH	TCP	22	배스천 서브넷(클라우드 배포용)
인바운드	HTTPS	TCP	443	웹 계층 서브넷
아웃바운드	모든 트래픽	전체	전체	

데이터 계층

데이터 서브넷은 외부 PostgreSQL 데이터베이스 서버가 상주하는 위치입니다.

트래픽	유형	프로토콜	포트 범위	원본
인바운드	SSH	TCP	22	배스천 서브넷(클라우드 배포용)
인바운드	PostgreSQL	TCP	5432	응용 프로그램 계층 서브넷
아웃바운드	모든 트래픽	전체	전체	

배스천

대부분의 기업 보안 팀은 온프레미스 관리 시스템에서 클라우드에 배포된 노드로의 직접 통신을 허용하지 않습니다. 대신, 클라우드 노드로 가는 모든 관리 SSH 트래픽에 대한 프록시로 배스천 호스트를 사용합니다("점프 서버"라고도 함). 클라우드 배포의 경우 참조 아키텍처의 모든 리소스에 대한 배스천 호스트 프록시 연결을 사용하는 것이 좋습니다. 온프레미스 환경에서 이 구성은 선택 사항입니다.

배스천 호스트는 관리 액세스를 인증하고 SSH 프로토콜을 통한 트래픽만 허용합니다.

트래픽	유형	프로토콜	포트 범위	원본	대상
인바운드	SSH	TCP	22	관리 컴퓨터 IP 주소	
아웃바운드	SSH	TCP	22		웹 계층 서브넷
아웃바운드	SSH	TCP	22		응용 프로그램 계층 서브넷

예: AWS에서 서브넷 및 보안 그룹 구성

이 섹션은 AWS의 Tableau Server 참조 아키텍처 배포에 대한 VPC 및 네트워크 환경을 만들고 구성하기 위한 단계별 절차를 제공합니다.

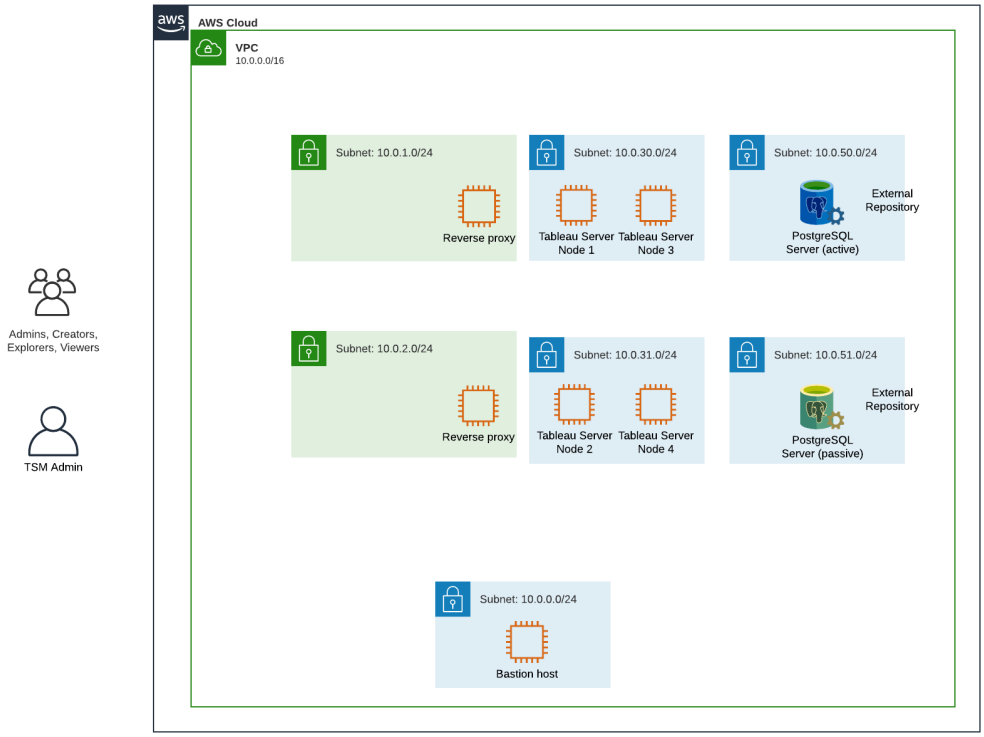
아래 슬라이드에서는 4개 계층의 참조 아키텍처를 보여 줍니다. 슬라이드를 진행하면서 구성 요소가 토폴로지 맵에 계층화됩니다.

1. VPC 서브넷 토폴로지 및 EC2 인스턴스: 배스천 호스트 1개, 역방향 프록시 서버 2개, Tableau Server 4개 및 PostgreSQL 서버 1개 이상
2. 프로토콜 흐름 및 인터넷 연결: 모든 인바운드 트래픽은 AWS 인터넷 게이트웨이를 통해 관리됩니다. 인터넷 트래픽은 NAT를 통해 라우팅됩니다.
3. 가용성 영역: 프록시, Tableau Server 및 PostgreSQL 호스트는 두 가용성 영역에 균등하게 배포됩니다.

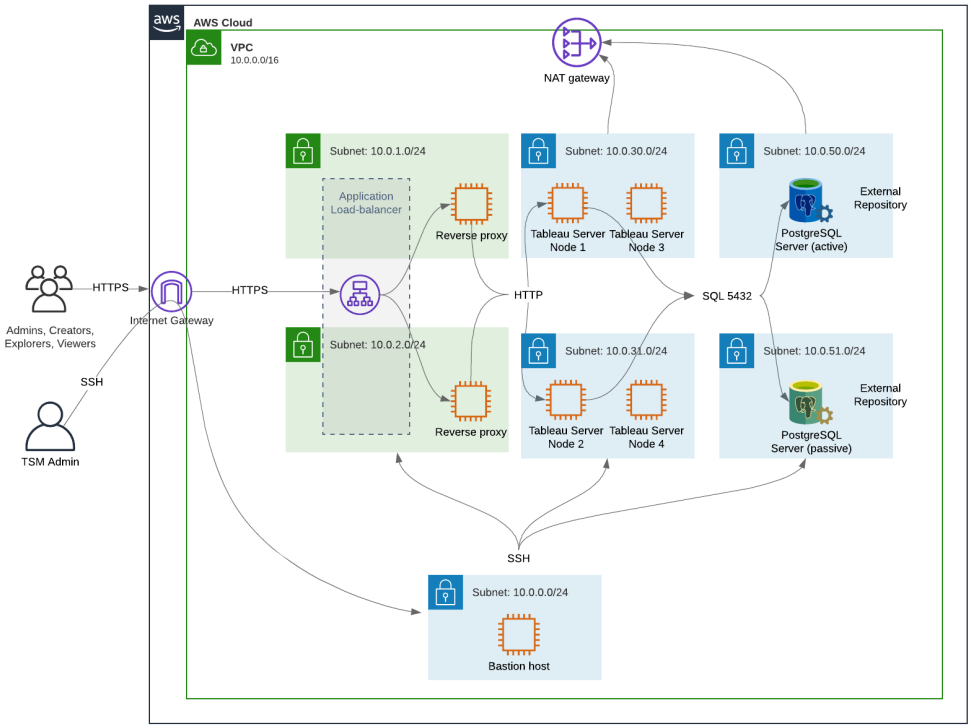
4. 보안 그룹: 네 개의 보안 그룹(공용, 사설, 데이터 및 배스천)이 프로토콜 수준에서 각 계층을 보호합니다.

AWS 참조 아키텍처

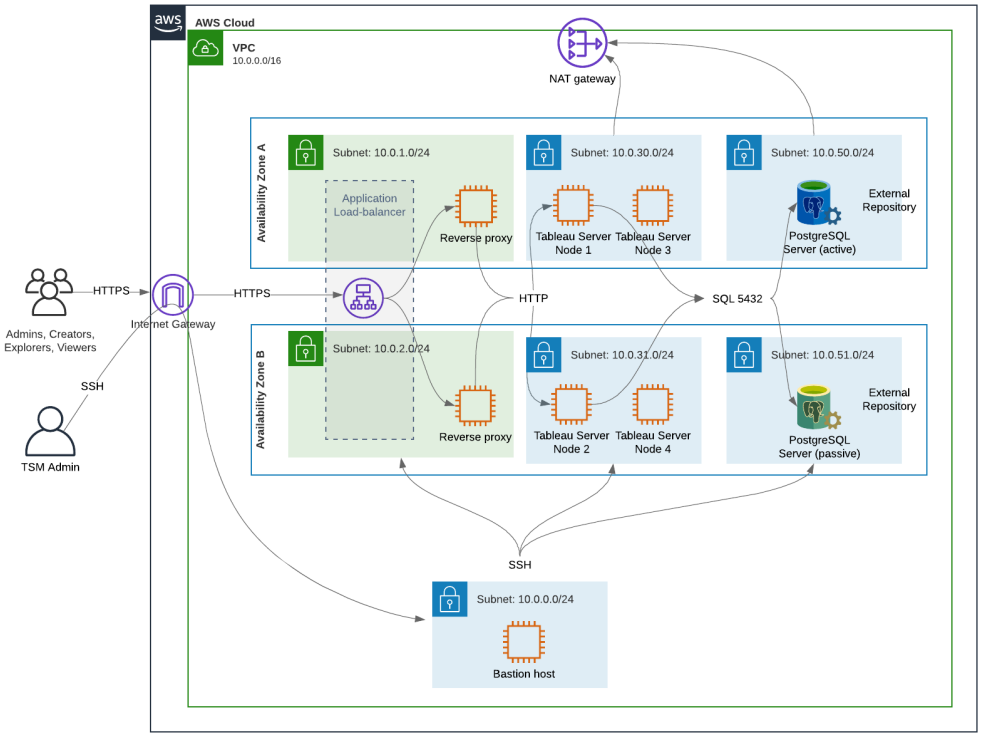
슬라이드 1: VPC 서브넷 토폴로지 및 EC2 인스턴스



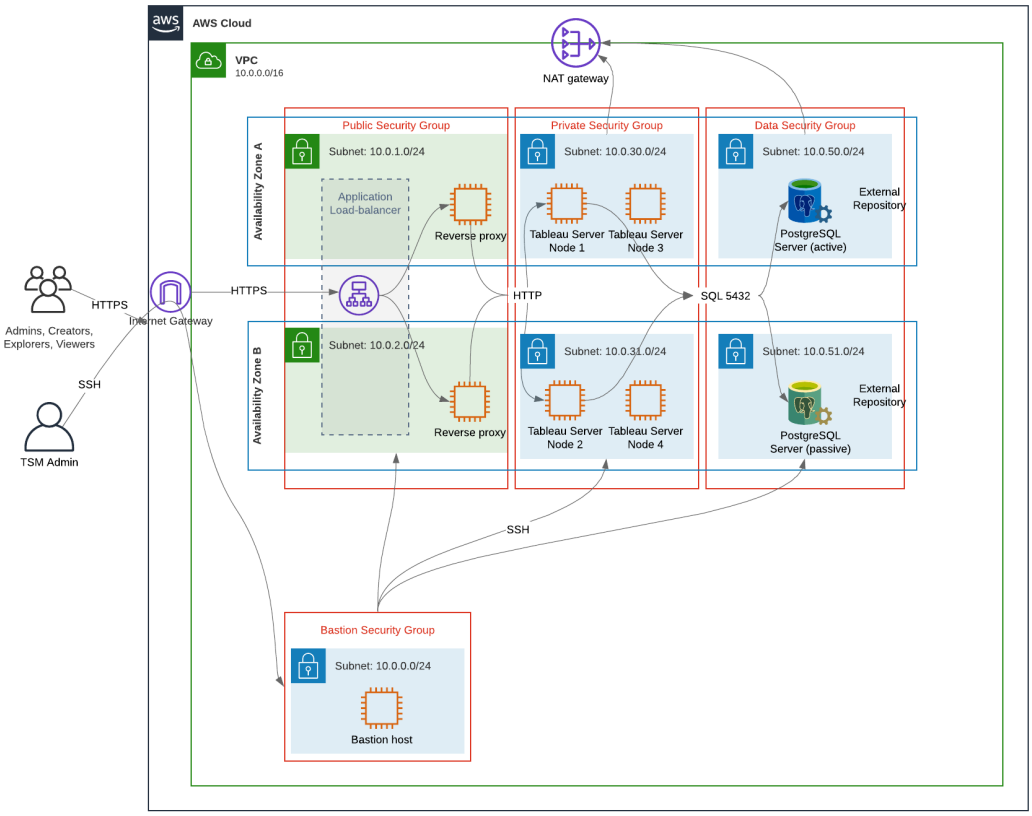
슬라이드 2: 프로토콜 흐름 및 연결



슬라이드 3: 가용성 영역


AWS Cloud
VPC
10.0.0.0/16
NAT gateway
Availability Zone A
Subnet: 10.0.1.0/24
Application
Load-balancer
Reverse proxy
Subnet: 10.0.30.0/24
Tableau Server
Node 1
Tableau Server
Node 3
Subnet: 10.0.50.0/24
External
Repository
PostgreSQL
Server (active)
HTTPS
Admins, Creators,
Explorers, Viewers
Internet Gateway
HTTPS
HTTP
SQL 5432
SSH
TSM Admin
Availability Zone B
Subnet: 10.0.2.0/24
Reverse proxy
Subnet: 10.0.31.0/24
Tableau Server
Node 2
Tableau Server
Node 4
Subnet: 10.0.51.0/24
External
Repository
PostgreSQL
Server (passive)
SSH
Subnet: 10.0.0.0/24
Bastion host

슬라이드 4: 보안 그룹



AWS 가용성 영역 및 고가용성

이 가이드에 나온 참조 아키텍처는 단일 호스트에 장애가 발생할 경우 중복성을 통해 가용성을 제공하는 배포를 명시합니다. 그러나 두 가용성 영역에 참조 아키텍처를 배포하는 AWS 사례에서는 가용성 영역에 장애가 발생하는 매우 드문 경우에만 가용성이 손상됩니다.

VPC 구성

이 섹션에서는 다음을 수행하는 방법을 설명합니다.

- VPC 설치 및 구성
- 인터넷 연결 구성

- 서브넷 구성
- 보안 그룹 생성 및 구성

VPC 구성

이 섹션의 절차는 클래식 VPC 경험의 UI에 매핑됩니다. AWS VPC 대시보드의 왼쪽 위에서 새 VPC 경험을 해제하여 클래식 뷰를 표시하도록 UI를 전환할 수 있습니다.

VPC 마법사를 실행하여 기본적인 사설 및 공용 서브넷과 기본적인 라우팅 및 네트워크 ACL을 만듭니다.

1. VPC를 구성하기 전에 탄력적 IP를 만들어야 합니다. 모든 기본값을 사용하여 할당을 만듭니다.
2. VPC 마법사 실행 > "공용 및 사설 서브넷이 있는 VPC"
3. 대부분의 기본값을 수락합니다. 다음은 예외입니다.
    - VPC 이름을 입력합니다.
    - 탄력적 IP 할당 ID를 지정합니다.
    - 다음 CIDR 마스크를 지정합니다.
        - 공용 서브넷의 IPv4 CIDR: 10.0.1.0/24, 이 서브넷의 이름을 `Public-a`로 바꿉니다.
        - 사설 서브넷의 IPv4 CIDR: 10.0.30.0/24, 이 서브넷의 이름을 `Private-a`로 바꿉니다.
    - 가용성 영역: 두 서브넷에 대해 현재 지역에 해당하는 **a** 옵션을 선택합니다.

    > **참고**: 이 예에서는 특정 AWS 데이터 센터의 가용성 영역을 서로 구분하기 위해 **a**와 **b**를 사용합니다. AWS에서 가용성 영역 이름이 여기에 표시된 예와 일치하지 않을 수 있습니다. 예를 들어 일부 가용성 영역에는 데이터 센터 내에 **c** 및 **d** 영역이 포함됩니다.

4. **VPC 만들기**를 클릭합니다.
5. VPC가 만들어지면 `Public-b`, `Private-b`, `Data` 및 `Bastion` 서브넷을 만듭니다. 서브넷을 만들려면 **서브넷 > 서브넷 만들기**를 클릭합니다.

    - `Public-b`: 가용성 영역의 경우 현재 지역에 해당하는 **b** 옵션을 선택합니다. CIDR 블록: 10.0.2.0/24

- `Private-b`: 가용성 영역의 경우 현재 지역에 해당하는 **b** 옵션을 선택합니다. CIDR 블록: 10.0.31.0/24
- `Data`: 가용성 영역의 경우 현재 지역에 해당하는 **a** 영역을 선택합니다. CIDR 블록: 10.0.50.0/24. 선택 사항: PostgreSQL 클러스터 전체의 외부 데이터베이스를 복제하려는 경우 CIDR 블록이 10.0.51.0/24인 가용성 영역 b에 Data-b 서브넷을 만듭니다.
- `Bastion`: 가용성 영역의 경우 영역 중 하나를 선택합니다. CIDR 블록: 10.0.0.0/24

6. 서브넷이 만들어지면 연결된 IGW(인터넷 게이트웨이)에 대해 구성된 라우팅 테이블을 사용하도록 공용 및 배스천 서브넷의 라우팅 테이블을 편집합니다. 그런 다음 NAT(네트워크 주소 변환기)에 대해 구성된 라우팅 테이블을 사용하도록 사설 및 데이터 서브넷을 편집합니다.
   - IGW 또는 NAT에 구성된 라우팅 테이블을 확인하려면 AWS 대시보드에서 **라우팅 테이블**을 클릭합니다. 두 라우팅 테이블 링크 중 하나를 선택하여 속성 페이지를 엽니다. **라우팅** > **대상** > **0.0.0.0/0**에서 대상 값을 확인합니다. 대상 값에 따라 라우팅 유형이 달라지며 `igw-` 또는 `nat-` 문자열로 시작됩니다.
   - 라우팅 테이블을 업데이트하려면 **VPC** > **서브넷** > [subnet_name] > **라우팅 테이블** > **라우팅 테이블 연결 편집**으로 이동합니다.

보안 그룹 구성

VPC 마법사는 단일 보안 그룹을 만들지만 이는 사용되지 않습니다. 다음 보안 그룹을 만듭니다(**보안 그룹** > **보안 그룹 만들기**). EC2 호스트는 위의 슬라이드 다이어그램과 같이 두 가용성 영역에 걸쳐 이러한 그룹에 설치됩니다.

- 새 보안 그룹 **사설**을 만듭니다. Tableau Server의 노드 4개가 모두 여기에 설치됩니다. 설치 프로세스의 나중에 사설 보안 그룹은 10.0.30.0/24 및 10.0.31.0/24 서브넷에 연결됩니다.
- 새 보안 그룹 **공용**을 만듭니다. 여기에는 프록시 서버가 설치됩니다. 설치 프로세스의 나중에 공용 보안 그룹은 10.0.1.0/24 및 10.0.2.0/24 서브넷에 연결됩니다.
- 새 보안 그룹 **데이터**를 만듭니다. PostgreSQL용 외부 Tableau 리포지토리가 모두 여기에 설치됩니다. 설치 프로세스의 나중에 데이터 보안 그룹은 10.0.50.0/24(선택적으로 10.0.51.0/24) 서브넷에 연결됩니다.
- 새 보안 그룹 **배스천**을 만듭니다. 여기에는 배스천 호스트가 설치됩니다. 설치 프로세스의 나중에 배스천 보안 그룹은 및 10.0.0.0/24 서브넷에 연결됩니다.

인바운드 및 아웃바운드 규칙 지정

AWS에서 보안 그룹은 온프레미스 환경의 방화벽과 같은 역할을 합니다. 보안 그룹의 송/수신에 허용되는 트래픽 유형(예: https, https 등), 프로토콜(TCP 또는 UDP) 및 포트 또는 포트 범위(예: 80, 443 등)를 지정해야 합니다. 각 프로토콜에 대해 대상 또는 원본 트래픽도 지정해야 합니다.

공용 보안 그룹 규칙

인바운드 규칙			
유형	**프로토콜**	**포트 범위**	**원본**
HTTP	TCP	80	0.0.0.0/0
HTTPS	TCP	443	0.0.0.0/0
SSH	TCP	22	배스천 보안 그룹

아웃바운드 규칙			
유형	**프로토콜**	**포트 범위**	**대상**
모든 트래픽	전체	전체	0.0.0.0/0

사설 보안 그룹 규칙

사설 보안 그룹에는 공용 보안 그룹의 HTTP 트래픽을 허용하는 인바운드 규칙이 포함됩니다. 연결을 확인하기 위해 배포 프로세스 중에만 HTTP 트래픽을 허용합니다. 역방향 프록시 배포를 마치고 Tableau에 대한 SSL을 구성한 후에는 HTTP 인바운드 규칙을 제거하는 것이 좋습니다.

인바운드 규칙			
유형	**프로토콜**	**포트 범위**	**원본**
HTTP	TCP	80	공용 보안 그룹

HTTPS	TCP	443	공용 보안 그룹
PostgreSQL	TCP	5432	데이터 보안 그룹
SSH	TCP	22	배스천 보안 그룹
모든 트래픽	전체	전체	사설 보안 그룹

아웃바운드 규칙			
유형	**프로토콜**	**포트 범위**	**대상**
모든 트래픽	전체	전체	0.0.0.0/0
PostgreSQL	TCP	5432	데이터 보안 그룹
SSH	TCP	22	배스천 보안 그룹

데이터 보안 그룹 규칙

인바운드 규칙			
유형	**프로토콜**	**포트 범위**	**원본**
PostgreSQL	TCP	5432	사설 보안 그룹
SSH	TCP	22	배스천 보안 그룹

아웃바운드 규칙			
유형	**프로토콜**	**포트 범위**	**대상**
모든 트래픽	전체	전체	0.0.0.0/0
PostgreSQL	TCP	5432	사설 보안 그룹
SSH	TCP	22	배스천 보안 그룹

배스천 호스트 보안 그룹 규칙

인바운드 규칙			
유형	**프로토콜**	**포트 범위**	**원본**
SSH	TCP	22	AWS(관리 컴퓨터)에 로그인하는 데 사용할 컴퓨터의 IP 주소 및 넷 마스크입니다.
SSH	TCP	22	사설 보안 그룹
SSH	TCP	22	공용 보안 그룹

아웃바운드 규칙			
유형	**프로토콜**	**포트 범위**	**대상**
SSH	TCP	22	AWS(관리 컴퓨터)에 로그인하는 데 사용할 컴퓨터의 IP 주소 및 넷 마스크입니다.
SSH	TCP	22	사설 보안 그룹
SSH	TCP	22	공용 보안 그룹
SSH	TCP	22	데이터 보안 그룹
HTTPS	TCP	443	0.0.0.0/0(선택 사항: 배스천 호스트에서 지원 소프트웨어를 다운로드하기 위해 인터넷에 연결해야 하는 경우 이 규칙을 만들어야 함)

공용 IP 자동 할당 사용

프록시 서버 및 배스천 호스트에 연결하기 위한 IP 주소를 제공합니다.

공용 및 배스천 서브넷의 경우:

1. 서브넷을 선택합니다.
2. **동작** 메뉴에서 "IP 자동 할당 설정 수정"을 선택합니다.
3. "공용 IPv4 주소 자동 할당 사용"을 클릭합니다.
4. **저장**을 클릭합니다.

부하 분산 장치

> **참고**: 이 가이드의 예제 배포에 따라 AWS에 설치하는 경우 배포 프로세스의 나중에 5부 - 웹 계층 구성에 설명된 대로 AWS 부하 분산 장치를 설치하고 구성해야 합니다.

온프레미스 배포의 경우 네트워크 관리자와 협력하여 참조 아키텍처의 웹 계층을 지원할 부하 분산 장치를 배포하십시오.

- Tableau 클라이언트의 HTTPS 요청을 수락하고 역방향 프록시 서버로 전달하는 웹용 응용 프로그램 부하 분산 장치
- 역방향 프록시:
  - 중복성을 구현하고 클라이언트 부하를 처리할 수 있도록 최소 2개의 프록시 서버를 구성하는 것이 좋습니다.
  - 부하 분산 장치의 HTTPS 트래픽을 수신합니다.
  - Tableau 호스트에 대한 고정 세션을 지원합니다.
  - 게이트웨이 프로세스를 실행하는 각 Tableau Server에 대한 라운드 로빈 부하 분산을 위한 프록시를 구성합니다.
  - 외부 IdP의 인증 요청을 처리합니다.
- 정방향 프록시: 라이선스 및 맵 기능을 위해 Tableau Server에서 인터넷에 액세스할 수 있어야 합니다. 사용하는 정방향 프록시 환경에 따라 Tableau Service URL에 대한 정방향 프록시 허용 목록을 구성해야 할 수 있습니다. *인터넷 통신*(Linux)을 참조하십시오.

호스트 컴퓨터 구성

최소 권장 하드웨어

다음 권장 사항은 참조 아키텍처에서 실제 데이터로 수행한 테스트 결과를 기준으로 합니다.

응용 프로그램 서버:

- CPU: 물리적 코어 8개(16vCPU)
- RAM: 128GB(16 GB/물리적 코어)
- 디스크 공간: 100 GB

데이터 서버

- CPU: 물리적 코어 8개(16vCPU)
- RAM: 128GB(16 GB/물리적 코어)
- 디스크 공간: 1TB. 배포에서 Tableau 파일 저장소에 대해 외부 저장소를 사용할 경우 적절한 디스크 공간을 계산해야 합니다. *Tableau Server에 외부 파일 저장소 설치*(Linux)를 참조하십시오.

프록시 서버

- CPU: 물리적 코어 2개(4vCPU)
- RAM: 8 GB(4 GB/물리적 코어)
- 디스크 공간: 100 GB

외부 리포지토리 데이터베이스

- CPU: 물리적 코어 8개(16vCPU)
- RAM: 128GB(16 GB/물리적 코어)
- 디스크 공간 요구 사항은 데이터 부하 및 백업에 미치는 영향에 따라 다릅니다. *디스크 공간 요구 사항(Linux)* 항목에서 *백업 및 복원 프로세스* 섹션을 참조하십시오.

디렉터리 구조

이 참조 아키텍처는 Tableau Server 패키지 및 데이터를 기본이 아닌 위치에 설치할 것을 권장합니다.

- `/app/tableau_server`에 패키지 설치: Tableau Server 패키지를 설치하기 전에 이 디렉터리 경로를 만든 다음 설치 중에 이 경로를 지정합니다.
- `/data/tableau_data`에 Tableau 데이터를 설치합니다. Tableau Server를 설치하기 전에 이 디렉터리를 만들지 마십시오. 대신, 설치 중에 경로를 지정한 다음 Tableau 설치 프로그램을 통해 경로를 만들고 적절한 사용 권한을 설정합니다.

구현 세부 정보는 설치 패키지 실행 및 TSM 초기화를 참조하십시오.

예: AWS에서 호스트 컴퓨터 설치 및 준비

이 섹션에서는 Tableau Server 참조 아키텍처의 각 서버 유형에 대한 EC2 호스트를 설치하는 방법을 설명합니다.

이 참조 아키텍처에는 8개의 호스트가 필요합니다.

- Tableau Server용 인스턴스 4개
- 프록시 서버용 인스턴스 2개 (Apache)
- 배스천 호스트용 인스턴스 1개
- EC2 PostgreSQL 데이터베이스 인스턴스 1개 또는 2개

호스트 인스턴스 세부 정보

아래의 세부 정보에 따라 호스트 컴퓨터를 설치합니다.

Tableau Server

- Amazon Linux 2
- 인스턴스 유형: m5a.8xlarge
- 보안 그룹 ID: 사설
- 저장소: EBS, 150GiB, gp2 볼륨 유형. 배포에서 Tableau 파일 저장소에 대해 외부 저장소를 사용할 경우 적절한 디스크 공간을 계산해야 합니다. *Tableau Server에 외부 파일 저장소 설치*(Linux)를 참조하십시오.

- 네트워크: 각 사설 서브넷(10.0.30.0/24 및 10.0.31.0/24)에 두 개의 EC2 호스트 설치
- Tableau 다운로드 페이지에서 최신 Tableau Server 2021.2 이상 rpm 패키지의 최신 유지 관리 버전을 각 Tableau 호스트에 복사합니다.

배스천 호스트

- Amazon Linux 2
- 인스턴스 유형: t3.micro
- 보안 그룹 ID: 배스천
- 저장소: EBS, 50GiB, gp2 볼륨 유형
- 네트워크: 배스천 서브넷 10.0.0.0/24

Tableau Server 독립 게이트웨이

- Amazon Linux 2
- 인스턴스 유형: t3.xlarge
- 보안 그룹 ID: 공용
- 저장소: EBS, 100GiB, gp2 볼륨 유형
- 네트워크: 각 공용 서브넷(10.0.1.0/24 및 10.0.2.0/24)에 한 개의 EC2 인스턴스 설치

PostgreSQL EC2 호스트

- Amazon Linux 2
- 인스턴스 유형: r5.4xlarge
- 보안 그룹 ID: 데이터
- 저장소: 디스크 공간 요구 사항은 데이터 부하 및 백업에 미치는 영향에 따라 다릅니다. *디스크 공간 요구 사항(Linux)* 항목에서 *백업 및 복원 프로세스* 섹션을 참조하십시오.
- 네트워크: 데이터 서브넷 10.0.50.0/24 (HA 클러스터에서 PostgreSQL을 복제하는 경우 10.0.51.0/24 서브넷에 두 번째 호스트 설치)

확인: VPC 연결

호스트 컴퓨터를 설치한 후 네트워크 구성을 확인합니다. 배스천 보안 그룹의 호스트에서 SSH를 통해 각 서브넷의 호스트에 연결하여 호스트 간 연결성을 확인합니다.

예: AWS의 배스천 호스트에 연결

1. SSH 에이전트로 사용할 관리 컴퓨터를 설정합니다. 이렇게 하면 개인 키 파일을 EC2 인스턴스에 배치하지 않고도 AWS의 호스트에 연결할 수 있습니다.

   Mac에서 SSH 에이전트를 구성하려면 다음 명령을 실행합니다.

   `ssh-add -K myPrivateKey.pem` 또는 최신 Mac OS의 경우 `ssh-add --apple-use-keychain myPrivateKey.pem`

   Windows의 경우 사설 Amazon VPC에서 실행되는 Linux 인스턴스에 안전하게 연결(영문) 항목을 참조하십시오.

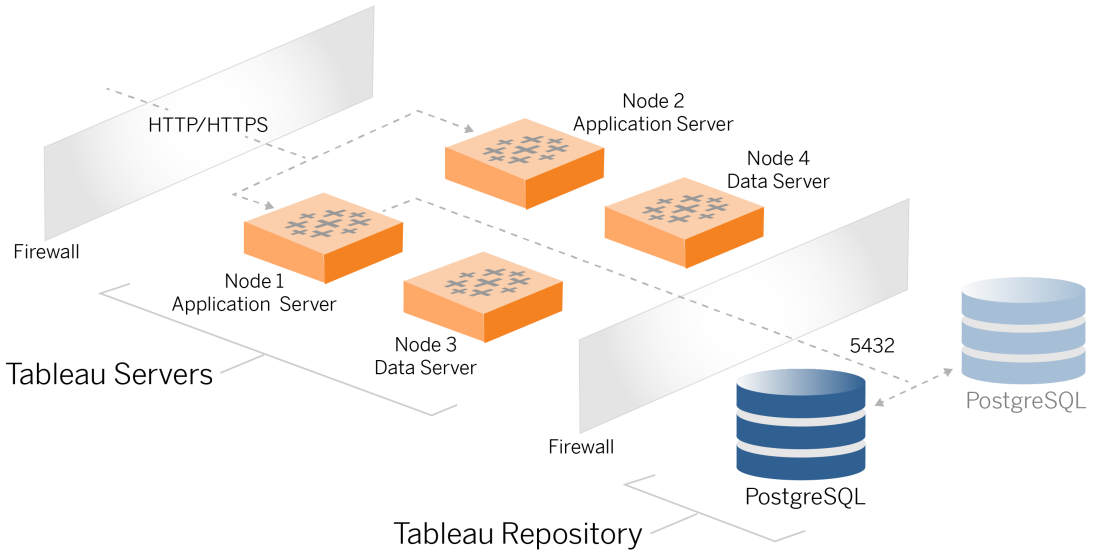
2. 다음 명령을 실행하여 배스천 호스트에 연결합니다.

   ```
   ssh -A ec2-user@<public-IP>
   ```

3. 그런 다음 배스천 호스트에서 사설 IP 주소를 사용하여 VPC의 다른 호스트에 연결할 수 있습니다. 예를 들면 다음과 같습니다.

   ```
   ssh -A ec2-user@10.0.1.93
   ```

4부 - Tableau Server 설치 및 구성



이 항목은 기준 Tableau Server 배포의 설치 및 구성을 완료하는 방법을 설명합니다. 이 절차는 AWS 및 Linux 참조 아키텍처의 예를 계속 따릅니다.

설치 절차 전반에 걸친 Linux 예는 RHEL 유사 배포에 대한 명령을 보여줍니다. 특히 여기에 나온 명령은 Amazon Linux 2 배포를 통해 개발되었습니다. Ubuntu 배포를 실행 중인 경우 그에 따라 명령을 편집합니다.

시작하기 전에

3부 - Tableau Server 엔터프라이즈 배포 준비에 설명된 대로 환경을 준비하고 검증해야 합니다.

PostgreSQL 설치, 구성 및 tar 백업 만들기

이 PostgreSQL 인스턴스는 Tableau Server 배포를 위한 외부 리포지토리를 호스팅합니다. Tableau를 설치하기 전에 PostgreSQL을 설치하고 구성해야 합니다.

Amazon RDS 또는 EC2 인스턴스에서 PostgreSQL을 실행할 수 있습니다. RDS에서 리포지토리를 실행하는 경우와 EC2 인스턴스에서 리포지토리를 실행하는 경우 간의 차이점에 대한 자세한 내용은 *Tableau Server 외부 리포지토리*(Linux)를 참조하십시오.

예를 들어 아래의 절차는 Amazon EC2 인스턴스에 Postgres를 설치하고 구성하는 방법을 보여줍니다. 여기에 나온 예제는 참조 아키텍처의 PostgreSQL에 대한 일반적인 설치 및 구성입니다. 따라서 조직의 DBA가 데이터 크기 및 성능 요구 사항에 따라 PostgreSQL 배포를 최적화해야 합니다.

요구 사항: PostgreSQL 1.6을 실행하고 uuid-ossp 모듈을 설치해야 합니다.

PostgreSQL 버전 관리

Tableau Server 외부 리포지토리로 사용할 호환되는 주 버전의 PostgreSQL을 설치해야 합니다. 또한 부 버전도 최소 요구 사항을 충족해야 합니다.

Tableau Server 버전	**PostgreSQL 최소 호환 버전**
2021.2.3 ~ 2021.2.8 2021.3.0 ~ 2021.3.7 2021.4.0 ~ 2021.4.3	12.6
2021.2.10 ~ 2021.2.14 2021.3.8 ~ 2021.3.13 2021.4.4 ~ 2021.4.8	12.8
2021.2.15 ~ 2021.2.16 2021.3.14 ~ 2021.3.15 2021.4.9 ~ 2021.4.10	12.10
2021.2.17 ~ 2021.2.18 2021.3.16 ~ 2021.3.17	12.11

2021.4.11 ~ 2021.4.12	
2021.3.26 2021.4.23	12.15
2022.1.0	13.3
2022.1.1 ~ 2022.1.3	13.4
2022.1.4 ~ 2022.1.6	13.6
2022.1.7 ~ 2022.1.16 2022.3.0 ~ 2022.3.7 2023.1.0 ~ 2023.1.4	13.7
2022.1.17 ~ 2022.1.19 2022.3.8 ~ 2022.3.11 2023.1.5 ~ 2023.1.7 2023.3.0 ~ 2023.3.3	13.11

PostgreSQL 설치

이 예제 설치 절차는 PostgreSQL 버전 13.6의 설치 방법을 설명합니다.

이전 부에서 만든 EC2 호스트에 로그인합니다.

1. 업데이트를 실행하여 최신 수정을 Linux OS에 적용합니다.

   ```
   sudo yum update
   ```

2. `/etc/yum.repos.d/` 경로에서 pgdg.repo. 파일을 만들고 편집합니다. 다음 구성 정보로 파일을 채웁니다.

```
[pgdg13]
name=PostgreSQL 13 for RHEL/CentOS 7 - x86_64

baseurl=https://download.postgresql.org/pub/repos/yum/13/redha-
t/rhel-7-x86_64
enabled=1
gpgcheck=0
```

3. Posgres 13.6을 설치합니다.

```
sudo yum install postgresql13-server-13.6-1PGDG.rhel7.x86_64
```

4. uuid-ossp 모듈을 설치합니다.

```
sudo yum install postgresql13-contrib-13.6-1PGDG.rhel7.x86_64
```

5. Postgres를 초기화합니다.

```
sudo /usr/pgsql-13/bin/postgresql-13-setup initdb
```

Postgres 구성

Postgres를 구성하여 기본 설치를 완료합니다.

1. 다음 두 항목으로 pg_hba 구성 파일(`/var/lib/pgsql/13/data/pg_hba.conf`)을 업데이트합니다. 각 항목에 Tableau Server를 실행할 서브넷의 마스크를 포함해야 합니다.

```
host all all 10.0.30.0/24 password

host all all 10.0.31.0/24 password
```

2. 다음 줄을 추가하여 PostgreSQL 파일(`/var/lib/pgsql/13/data/postgresql.conf`)을 업데이트합니다.

```
listen_addresses = '*'
```

3. 재부팅 시 Postgres를 시작하도록 구성합니다.

```
sudo systemctl enable --now postgresql-13
```

4. 슈퍼 사용자 비밀번호를 설정합니다.

```
sudo su - postgres

psql -c "alter user postgres with password 'StrongPassword'"
```

> **참고**: 강력한 비밀번호를 설정하십시오. 여기 예제에 표시된 대로 `'StrongPassword'`를 사용하지 마십시오.

```
exit
```

5. Postgres를 다시 시작합니다.

```
sudo systemctl restart postgresql-13
```

PostgreSQL 1단계 tar 백업 만들기

PostgreSQL 구성의 tar 백업을 만듭니다. 현재 구성의 tar 스냅샷을 만들면 배포를 계속하는 동안 오류가 발생할 경우 시간이 절약됩니다.

이 백업을 "1단계" 백업이라고 합니다.

PostgreSQL 호스트에서

1. Postgres 데이터베이스 인스턴스를 중지합니다.

```
sudo systemctl stop postgresql-13
```

2. 다음 명령을 실행하여 tar 백업을 만듭니다.

```
sudo su

cd /var/lib/pgsql

tar -cvf step1.13.bkp.tar 13
```

```
exit
```

3. Postgres 데이터베이스를 시작합니다.

```
sudo systemctl start postgresql-13
```

1단계 복원

설치하는 동안 Tableau Server 초기 노드가 실패하면 1단계로 복원합니다.

1. Tableau를 실행하는 컴퓨터에서 obliterate 스크립트를 실행하여 호스트에서 Tableau Server를 완전히 제거합니다.

```
sudo /app/tableau_server/packages/scripts.<version_
code>/./tableau-server-obliterate -a -y -y -y -l
```

2. PostgreSQL 1단계 tar를 복원합니다. Postgres를 실행하는 컴퓨터에서 다음 명령을 실행합니다.

```
sudo su
systemctl stop postgresql-13
cd /var/lib/pgsql
tar  -xvf step1.13.bkp.tar
systemctl start postgresql-13
exit
```

Tableau Server의 초기 노드를 설치하는 설치 프로세스를 다시 시작합니다.

설치 전 수행할 작업

이 가이드에 설명된 AWS/Linux 구현 예에 따라 Tableau를 배포하는 경우 자동 설치 스크립트인 TabDeploy4EDG를 실행할 수 있습니다. TabDeploy4EDG 스크립트는 다음 절

차에 설명된 4노드 Tableau 배포의 예시 설치를 자동화합니다. 부록 - AWS 배포 도구 상자를 참조하십시오.

Tableau Server의 초기 노드 설치

다음 절차에서는 참조 아키텍처에 정의된 대로 Tableau Server의 초기 노드를 설치하는 방법에 대해 설명합니다. 패키지 설치와 TSM 초기화를 제외하고, 가능한 경우 다음 절차에서 TSM 명령줄을 사용합니다. TSM CLI를 사용하면 플랫폼에 구애받지 않을 뿐 아니라 가상화 및 헤드리스 환경에 보다 원활하게 설치할 수 있습니다.

설치 패키지 실행 및 TSM 초기화

노드 1 호스트 서버에 로그인합니다.

1. 업데이트를 실행하여 최신 수정을 Linux OS에 적용합니다.

   ```
   sudo yum update
   ```

2. Tableau 다운로드 페이지에서 Tableau Server를 실행할 호스트 컴퓨터로 설치 패키지를 복사합니다.

   예를 들어 Linux RHEL 유사 운영 체제를 실행하는 컴퓨터에서 다음을 실행합니다.

   ```
   wget
   https://downloads.tableau.com/esdalt/2022<version>/tableau-
   server-<version>.rpm
   ```

   여기서 `<version>`은 릴리스 번호입니다.

3. 종속 항목을 다운로드 및 설치합니다.

   ```
   sudo yum deplist tableau-server-<version>.rpm | awk
   '/provider:/ {print $2}' | sort -u | xargs sudo yum -y install
   ```

4. 루트 디렉터리에 `/app/tableau_server` 경로를 만듭니다.

```
sudo mkdir -p /app/tableau_server
```

5. 설치 프로그램을 실행하고 `/app/tableau_server` 설치 경로를 지정합니다. 예를 들어 Linux RHEL과 같은 운영 체제에서 다음을 실행합니다.

```
sudo rpm -i --prefix /app/tableau_server tableau-server-
<version>.x86_64.rpm
```

6. `/app/tableau_server/packages/scripts.<version_code>/` 디렉터리로 변경하고 거기에 있는 `initialize-tsm` 스크립트를 실행합니다.

```
sudo ./initialize-tsm -d /data/tableau_data --accepteula
```

7. 초기화가 완료되면 셸을 끝냅니다.

```
exit
```

Tableau Server 활성화 및 등록

1. 노드 1 호스트 서버에 로그인합니다.

2. 이 단계에서 Tableau Server 제품 키를 입력합니다. 구매한 각 라이선스 키에 대해 다음 명령을 실행합니다.

```
tsm licenses activate -k <product key>
```

3. 다음과 같은 형식으로 json 등록 파일을 만듭니다.

```
{
"zip" : "97403",
"country" : "USA",
"city" : "Springfield",
"last_name" : "Simpson",
"industry" : "Energy",
"eula" : "yes",
"title" : "Safety Inspection Engineer",
"company_employees" : "100",
```

```
"phone" : "5558675309",
"company" : "Example",
"state" : "OR",
"opt_in" : "true",
"department" : "Engineering",
"first_name" : "Homer",
"email" : "homer@example.com"
}
```

4. 파일의 변경 내용을 저장한 후 `--file` 옵션을 사용하여 전달하고 Tableau Server를 등록합니다.

   ```
   tsm register --file path_to_registration_file.json
   ```

ID 저장소 구성

> **참고**: 배포에서 Tableau 파일 저장소에 외부 저장소를 사용할 경우 ID 저장소를 구성하기 전에 외부 파일 저장소를 사용하도록 설정해야 합니다. *Tableau Server에 외부 파일 저장소 설치*(Linux)를 참조하십시오.

기본 참조 아키텍처에서는 로컬 ID 저장소를 사용합니다. `tsm settings import` 명령으로 `config.json` 파일을 전달하여 초기 호스트를 로컬 ID 저장소로 구성합니다.

운영 체제에 따라 `config.json` 파일을 가져옵니다.

`config.json` 파일은 scripts.<version> 디렉터리 경로(예: `scripts.20204.21.0217.1203`)에 포함되어 있으며 ID 저장소를 구성할 수 있도록 형식이 지정됩니다.

다음 명령을 실행하여 `config.json` 파일을 가져옵니다.

```
tsm settings import -f /app/tableau_
server/packages/scripts.<version_code>/config.json
```

외부 Postgres 구성

1. 다음 구성 설정을 사용하여 외부 데이터베이스 json 파일을 만듭니다.

```
{
"flavor":"generic",
"masterUsername":"postgres",
"host":"<instance ip address>",
"port":5432
}
```

2. 파일에 변경 내용을 저장한 후 다음 명령을 사용하여 파일을 전달합니다.

```
tsm topology external-services repository enable -f
<filename>.json --no-ssl
```

Postgres 마스터 사용자 이름 및 비밀번호를 묻는 메시지가 표시됩니다.

`--no-ssl` 옵션은 Postgres 서버가 SSL/TLS에 대해 구성된 경우에만 SSL/TLS를 사용하도록 Tableau를 구성합니다. Postgres가 SSL/TLS에 대해 구성되지 않은 경우 연결이 암호화되지 않습니다. 6부 - 설치 후 구성 에서는 첫 번째 배포 단계를 완료한 후 Postgres 연결에 SSL/TLS를 사용하도록 설정하는 방법에 대해 설명합니다.

3. 변경 내용을 적용합니다.

이 명령을 실행하여 변경 사항을 적용하고 Tableau Server를 다시 시작합니다.

```
tsm pending-changes apply
```

4. 1단계에서 사용한 구성 파일을 삭제합니다.

노드 1 설치 완료

1. Tableau Server가 설치되면 서버를 초기화해야 합니다.

다음 명령을 실행합니다.

```
tsm initialize --start-server --request-timeout 1800
```

2. 초기화가 완료되면 Tableau Server 관리자 계정을 만들어야 합니다.

   TSM 운영 체제 구성 요소를 설치 및 관리하는 데 사용하는 컴퓨터 계정과 달리 Tableau Server 관리자 계정은 Tableau Server 사용자, 프로젝트 및 사이트를 만드는 데 사용되는 응용 프로그램 계정입니다. Tableau Server 관리자는 Tableau 리소스에 대한 사용 권한도 적용합니다. 다음 명령을 실행하여 초기 관리자 계정을 만듭니다. 다음 예에서는 사용자를 `tableau-admin`이라고 합니다.

   ```
   tabcmd initialuser --server http://localhost --
   username "tableau-admin"
   ```

   Tabcmd는 이 사용자의 비밀번호를 설정하라는 메시지를 표시합니다.

확인: 노드 1 구성

1. 다음 명령을 실행하여 TSM 서비스가 실행되고 있는지 확인합니다.

   ```
   tsm status -v
   ```

   Tableau에서 다음을 반환합니다.

   ```
   external:
   Status: RUNNING
   'Tableau Server Repository 0' is running (Active Repository).
   node1: localhost
   Status: RUNNING
   'Tableau Server Gateway 0' is running.
   'Tableau Server Application Server 0' is running.
   'Tableau Server Interactive Microservice Container 0' is
   running.
   'MessageBus Microservice 0' is running.
   'Relationship Query Microservice 0' is running.
   'Tableau Server VizQL Server 0' is running.
   ...
   ```

   모든 서비스가 나열됩니다.

2. 다음 명령을 실행하여 Tableau 관리 사이트가 실행되고 있는지 확인합니다.

```
curl localhost
```

처음 몇 개 줄은 Vizportal html을 보여주며 다음과 유사합니다.

```
<!DOCTYPE html>
<html xmlns:ng="" xmlns:tb="">
<head ng-csp>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="initial-scale=1, maximum-
scale=2, width=device-width, height=device-height, viewport-
fit=cover">
<meta name="format-detection" content="telephone=no">
<meta name="vizportal-config ...
```

Step 2 tar 백업

초기 설치를 확인한 후 두 개의 tar 백업을 수행합니다.

- PostgreSQL
- Tableau 초기 노드(노드 1)

대부분의 경우 이러한 tar 파일을 복원하면 초기 노드 설치를 복구할 수 있습니다. tar 파일을 복원하는 것이 초기 노드를 다시 설치하고 다시 초기화하는 것보다 훨씬 빠릅니다.

Step 2 tar 파일 만들기

1. Tableau의 초기 노드에서 Tableau를 중지합니다.

```
tsm stop
```

Tableau가 멈출 때까지 기다린 후 다음 단계를 진행합니다.

2. PostgreSQL 호스트에서 Postgres 데이터베이스 인스턴스를 중지합니다.

```
sudo systemctl stop postgresql-13
```

3. 다음 명령을 실행하여 tar 백업을 만듭니다.

```
sudo su

cd /var/lib/pgsql

tar -cvf step2.13.bkp.tar 13

exit
```

4. Postgres tar 파일이 루트 권한으로 만들어졌는지 확인합니다.

```
sudo ls -al /var/lib/pgsql
```

5. Tableau 호스트에서 Tableau 관리 서비스를 중지합니다.

```
sudo /app/tableau_server/packages/scripts.<version_
code>/./stop-administrative-services
```

6. 다음 명령을 실행하여 tar 백업을 만듭니다.

```
cd /data

sudo tar -cvf step2.tableau_data.bkp.tar tableau_data
```

7. Postgres 호스트에서 Postgres 데이터베이스를 시작합니다.

```
sudo systemctl start postgresql-13
```

8. Tableau 관리 서비스를 시작합니다.

```
sudo /app/tableau_server/packages/scripts.<version_
code>/./start-administrative-services
```

9. `tsm status` 명령을 실행하여 다시 시작 전에 TSM 상태를 모니터링합니다.

   대부분의 경우 이 명령은 먼저 DEGRADED 또는 ERROR 상태를 반환합니다. 잠시 기다린 후 명령을 다시 실행합니다. ERROR 또는 DEGRADED 상태가 반환되면 계

속 기다립니다. STOPPED 상태가 반환될 때까지 TSM을 시작하려고 하지 마십시오. 그런 후 다음 명령을 실행합니다.

```
tsm start
```

Step 2 복원

이 프로세스는 Tableau 노드 1 및 Postgres 인스턴스를 Step 2로 복원합니다. 이 단계로 복원한 후 나머지 Tableau 노드를 다시 배포할 수 있습니다.

1. 초기 Tableau 호스트(노드 1)에서 tsm 서비스를 중지합니다.

   ```
   tsm stop
   ```

2. Tableau Server 배포의 모든 노드에서 Tableau 관리 서비스를 중지합니다. 각 노드에서 다음 명령을 순서대로 실행합니다(노드 1, 노드 2, 노드 3).

   ```
   sudo /app/tableau_server/packages/scripts.<version_
   code>/./stop-administrative-services
   ```

3. Tableau 서비스가 중지된 후 PostgreSQL Step 2 tar를 복원합니다. Postgres를 실행하는 컴퓨터에서 다음 명령을 실행합니다.

   - ```
     sudo su
     ```

     ```
     systemctl stop postgresql-13
     ```

     ```
     cd /var/lib/pgsql
     ```

     ```
     tar  -xvf step2.13.bkp.tar
     ```

     ```
     systemctl start postgresql-13
     ```

     ```
     exit
     ```

4. Tableau Step 2 tar를 복원합니다. 초기 Tableau 호스트에서 다음 명령을 실행합니다.
     ```

```
cd /data

sudo rm -rf tableau_data

sudo tar  -xvf step2.tableau_data.bkp.tar
```

5. Tableau 노드 1 컴퓨터에서 다음 파일을 제거합니다.

   - `sudo rm /data/tableau_data/data/tabsvc/appzookeeper/0/version-2/currentEpoch`
   - `sudo rm /data/tableau_data/data/tabsvc/appzookeeper/0/version-2/acceptedEpoch`
   - `sudo rm /data/tableau_data/data/tabsvc/tabadminagent/0/servicestate.json`

6. Tableau 관리 서비스를 시작합니다.

   ```
   sudo /app/tableau_server/packages/scripts.<version_code>/./start-administrative-services
   ```

7. Tableau Systmctl 파일을 다시 로드한 다음 `start-administrative-services`를 다시 실행합니다.

   ```
   sudo su -l tableau -c "systemctl --user daemon-reload"

   sudo /app/tableau_server/packages/scripts.<version_code>/./start-administrative-services
   ```

8. 노드 1에서 `tsm status` 명령을 실행하여 다시 시작 전에 TSM 상태를 모니터링합니다.

   일부 경우 `Cannot connect to server...` 오류가 표시됩니다. 이 오류는 tabadmincontroller 서비스가 다시 시작되지 않았기 때문에 발생합니다. 계속해서 `tsm status`를 주기적으로 실행합니다. 10분 후에도 이 오류가 사라지지 않으면 `start-administrative-services` 명령을 다시 실행합니다.

몇 분 후에 `tsm status` 명령은 DEGRADED 상태를 반환하고 그런 다음 ERROR 상태를 반환합니다. STOPPED 상태가 반환될 때까지 TSM을 시작하지 마십시오. 그런 후 다음 명령을 실행합니다.

```
tsm start
```

나머지 노드에서 Tableau Server를 설치하는 설치 프로세스를 다시 시작합니다.

나머지 노드에 Tableau Server 설치

배포를 계속하려면 각 노드에 Tableau 설치 프로그램을 복사합니다.

노드 구성 개요

이 섹션에서는 노드 2~4를 구성하는 프로세스를 설명합니다. 이후 섹션에는 각 단계의 자세한 구성 정보와 검증 절차가 제공됩니다.

Tableau Server의 2~4번 노드를 설치할 때는 노드 설치 중에 부트스트랩 파일을 생성, 복사 및 참조해야 합니다.

부트스트랩 파일을 생성하려면 초기 노드에서 TSM 명령을 실행합니다. 그런 다음 부트스트랩 파일을 대상 노드에 복사하고 대상 노드에서 노드 초기화의 일부로 실행합니다.

다음 json 콘텐츠는 부트스트랩 파일의 예를 보여줍니다. (인증서 및 암호화 관련 값은 예제 파일의 가독성을 높이기 위해 잘렸습니다.)

```
{
"initialBootstrapSettings" : {
       "certificate" : "-----BEGIN CERTIFICATE-----\r\...=\r\n-----END
CERTIFICATE-----",
       "port" : 8850,
       "configurationName" : "tabsvc",
       "clusterId" : "tabsvc-clusterid",
       "cryptoKeyStore" : "zs7OzgAAAAIAAAABAAAAA...w==",
       "toksCryptoKeystore" : "LS0tLS1CRUdJTiBUT00tLS0tCjM5MDBh...L",
```

```
        "sessionCookieMaxAge" : 7200,
        "nodeId" : "node1",
        "machineAddress" : "ip-10-0-1-93.us-west-1.compute.internal",
        "cryptoEnabled" : true,
        "sessionCookieUser" : "tsm-bootstrap-user",
        "sessionCookieValue" :
"eyJjdHkiOiJKV1QiLCJlbmMiOiJBMTI4Q0JDLUhQ...",
        "sessionCookieName" : "AUTH_COOKIE"
        }
}
```

부트스트랩 파일은 노드 1을 인증하기 위한 연결 기반 검증을 포함하며 부트스트랩 프로세스를 위한 암호화된 채널을 만듭니다. 부트스트랩 세션은 시간이 제한되어 있으며 노드 구성 및 검증에는 많은 시간이 소요됩니다. 그러므로 노드를 구성할 때 새 부트스트랩을 만들고 복사할 계획을 세우십시오.

부트스트랩 파일을 실행한 후 초기 Tableau Server 노드에 로그인하고 새 노드에 대한 프로세스를 구성합니다. 노드 구성이 완료되면 변경 내용을 적용하고 초기 노드를 다시 시작해야 합니다. 새 노드가 구성되고 시작됩니다. 노드를 추가하면 배포를 구성하고 다시 시작하는 시간이 계속해서 길어집니다.

설치 절차 전반에 걸친 Linux 예는 RHEL 유사 배포에 대한 명령을 보여줍니다. Ubuntu 배포를 실행 중인 경우 그에 따라 명령을 편집합니다.

1. 업데이트를 실행하여 최신 수정을 Linux OS에 적용합니다.

   ```
   sudo yum update
   ```

2. 종속 항목을 다운로드 및 설치합니다.

   ```
   sudo yum deplist tableau-server-<version>.rpm | awk
   '/provider:/ {print $2}' | sort -u | xargs sudo yum -y install
   ```

3. 루트 디렉터리에 `/app/tableau_server` 경로를 만듭니다.

   ```
   sudo mkdir -p /app/tableau_server
   ```

4. 설치 프로그램을 실행하고 /app/tableau_server 설치 경로를 지정합니다. 예를 들어 Linux RHEL과 같은 운영 체제에서 다음을 실행합니다.

```
sudo rpm -i --prefix /app/tableau_server tableau-server-
<version>.x86_64.rpm
```

부트스트랩 파일을 생성, 복사 및 사용하여 TSM 초기화

다음 절차는 다른 노드에서 TSM을 초기화할 때 부트스트랩 파일을 생성, 복사 및 사용하는 방법을 보여줍니다. 이 예에서 부트스트랩 파일의 이름은 boot.json입니다.

이 예에서 호스트 컴퓨터는 AWS에서 실행되고 EC2 호스트는 Amazon Linux 2를 실행합니다.

1. 초기 노드(노드 1)에 연결하고 다음 명령을 실행합니다.

```
tsm topology nodes get-bootstrap-file --file boot.json
```

2. 부트스트랩 파일을 노드 2에 복사합니다.

```
scp boot.json ec2-user@10.0.31.83:/home/ec2-user/
```

3. 노드 2에 연결하고 Tableau Server 스크립트 디렉터리로 전환합니다.

```
cd /app/tableau_server/packages/scripts.<version_number>
```

4. initialize-tsm 명령을 실행하고 부트스트랩 파일을 참조합니다.

```
sudo ./initialize-tsm -d /data/tableau_data -b /home/ec2-
user/boot.json --accepteula
```

5. initialize-tsm이 완료되면 boot.json을 삭제한 다음 세션을 끝내거나 로그아웃합니다.

프로세스 구성

Tableau Server 관리 컨트롤러(TSM 컨트롤러)가 실행 중인 노드에서 Tableau Server 클러스터를 구성해야 합니다. TSM 컨트롤러는 초기 노드에서 실행됩니다.

Process Status

The real-time status of processes running in Tableau Server.

Process	Node 1	Node 2	Node 3	Node 4	External Node
Cluster Controller	✓	✓	✓	✓	
Gateway	✓	✓			
Application Server	✓	✓			
VizQL Server	✓ ✓	✓ ✓			
Cache Server	✓ ✓	✓ ✓			
Search & Browse	✓	✓			
Backgrounder			✓ ✓ ✓ ✓	✓ ✓ ✓ ✓	
Data Server	✓ ✓	✓ ✓			
Data Engine	✓	✓	✓	✓	
File Store			✓	✓	
Repository					E
Tableau Prep Conductor			✓	✓	
Metrics	✓				

Refresh Status ✓ Active Busy Passive Unlicensed ✗ Down E External Status unavailable

노드 2 구성

1. 노드 2에서 부트스트랩 파일을 사용하여 TSM을 초기화한 후 초기 노드에 로그인합니다.

2. 초기 노드(`node1`)에서 다음 명령을 실행하여 노드 2의 프로세스를 구성합니다.

```
tsm topology set-process -n node2 -pr clustercontroller -c 1
tsm topology set-process -n node2 -pr gateway -c 1
tsm topology set-process -n node2 -pr vizportal -c 1
```

```
tsm topology set-process -n node2 -pr vizqlserver -c 2
tsm topology set-process -n node2 -pr cacheserver -c 2
tsm topology set-process -n node2 -pr searchserver -c 1
tsm topology set-process -n node2 -pr dataserver -c 2
tsm topology set-process -n node2 -pr clientfileservice -c 1
tsm topology set-process -n node2 -pr tdsservice -c 1
tsm topology set-process -n node2 -pr collections -c 1
tsm topology set-process -n node2 -pr contentexploration -c 1
```

버전 2022.1 이상을 설치하는 경우 인덱스 및 검색 서비스도 추가합니다.

```
tsm topology set-process -n node2 -pr indexandsearchserver -c 1
```

버전 2023.3 이상을 설치하는 경우 인덱스 및 검색 서비스만 포함하십시오. 검색 및 찾아보기(searchserver) 서비스는 추가하지 마십시오.

3. 적용하기 전에 구성을 검토합니다. 다음 명령을 실행합니다.

```
tsm pending-changes list
```

4. 변경 내용이 보류 목록(보류 목록에는 다른 서비스도 포함됨)에 있는지 확인한 후 변경 내용을 적용합니다.

```
tsm pending-changes apply
```

변경 내용을 적용하려면 다시 시작해야 합니다. 구성 및 다시 시작에는 시간이 조금 걸립니다.

5. 노드 2 구성을 확인합니다. 다음 명령을 실행합니다.

```
tsm status -v
```

노드 3 구성

노드 3에서 부트스트랩 프로세스를 사용하여 TSM을 초기화한 다음 아래의 `tsm topology set-process` 명령을 실행합니다.

프로세스를 설정할 때마다 조정 서비스 경고가 표시됩니다. 프로세스를 설정할 때는 이 경고를 무시해도 됩니다.

1. 노드 3에서 부트스트랩 파일을 사용하여 TSM을 초기화한 후 초기 노드(`node1`)에 로그인하고 다음 명령을 실행하여 프로세스를 구성합니다.

   ```
   tsm topology set-process -n node3 -pr clustercontroller -c 1
   tsm topology set-process -n node3 -pr clientfileservice -c 1
   tsm topology set-process -n node3 -pr backgrounder -c 4
   tsm topology set-process -n node3 -pr filestore -c 1
   ```

   버전 2022.1 이상을 설치하는 경우 인덱스 및 검색 서비스도 추가합니다.

   ```
   tsm topology set-process -n node3 -pr indexandsearchserver -c 1
   ```

2. 적용하기 전에 구성을 검토합니다. 다음 명령을 실행합니다.

   ```
   tsm pending-changes list
   ```

3. 변경 내용이 보류 목록(이 목록에는 자동으로 구성되는 다른 서비스도 포함됨)에 있는지 확인한 후 변경 내용을 적용합니다.

   ```
   tsm pending-changes apply --ignore-warnings
   ```

   변경 내용을 적용하려면 다시 시작해야 합니다. 구성 및 다시 시작에는 시간이 조금 걸립니다.

4. 다음 명령을 실행하여 구성을 확인합니다.

   ```
   tsm status -v
   ```

노드 1~3에 조정 서비스 집합 배포

표준 참조 아키텍처인 4노드 배포의 경우 다음 절차를 실행합니다.

1. 노드 1에서 다음 명령을 실행합니다.

   ```
   tsm stop
   tsm topology deploy-coordination-service -n node1,node2,node3
   ```

   이 프로세스에는 TSM을 재시작하는 작업이 포함되어 있으며, 시간이 다소 걸립니다.

2. 조정 서비스가 배포된 후 TSM을 시작합니다.

   ```
   tsm start
   ```

3단계 tar 백업 만들기

설치를 확인한 후 4개의 tar 백업을 수행합니다.

- PostgreSQL
- Tableau 초기 노드(노드 1)
- Tableau 노드 2
- Tableau 노드 3

Step 3 tar 파일 만들기

1. Tableau의 초기 노드에서 Tableau를 중지합니다.

   ```
   tsm stop
   ```

2. TSM이 중지된 후 각 노드에서 Tableau 관리 서비스를 중지합니다. 각 노드에서 다음 명령을 순서대로 실행합니다(노드 1, 노드 2, 노드 3).

   ```
   sudo /app/tableau_server/packages/scripts.<version_
   code>/./stop-administrative-services
   ```

3. PostgreSQL 호스트에서 Postgres 데이터베이스 인스턴스를 중지합니다.

```
sudo systemctl stop postgresql-12
```

4. 다음 명령을 실행하여 tar 백업을 만듭니다.

```
sudo su

cd /var/lib/pgsql

tar -cvf step3.12.bkp.tar 12

exit
```

5. Postgres tar 파일이 루트 권한으로 만들어졌는지 확인합니다.

```
sudo ls -al /var/lib/pgsql
```

6. Postgres 호스트에서 Postgres 데이터베이스를 시작합니다.

```
sudo systemctl start postgresql-12
```

7. 노드 1, 노드 2 및 노드 3에 tar 백업을 만듭니다. 각 노드에서 다음 명령을 실행합니다.

   - `cd /data`

     ```
     sudo tar -cvf step3.tableau_data.bkp.tar tableau_data
     ```

   - Tableau tar 파일이 루트 권한으로 만들어졌는지 확인합니다.

     ```
     ls -al
     ```

8. 각 노드에서 Tableau 관리 서비스를 순서대로 시작합니다(노드 1, 노드 2, 노드 3).

```
sudo /app/tableau_server/packages/scripts.<version_
code>/./start-administrative-services
```

9. `tsm status` 명령을 실행하여 다시 시작 전에 TSM 상태를 모니터링합니다.

   대부분의 경우 이 명령은 DEGRADED 상태를 반환한 다음 ERROR 상태를 반환합니다. 잠시 기다린 후 명령을 다시 실행합니다. ERROR 또는 DEGRADED 상태가

반환되면 계속 기다립니다. STOPPED 상태가 반환될 때까지 TSM을 시작하려고 하지 마십시오. 그런 후 다음 명령을 실행합니다.

```
tsm start
```

Step 3 복원

다음 프로세스는 Tableau 노드 1, 노드 2 및 노드 3을 복원합니다. 또한 Postgres 인스턴스를 Step 3으로 복원합니다. 이 Step으로 복원한 후 조정 서비스, 노드 4 및 최종 노드 구성을 배포할 수 있습니다.

1. 초기 Tableau 호스트(노드 1)에서 tsm 서비스를 중지합니다.

   ```
   tsm stop
   ```

2. TSM이 중지된 후 노드 1, 노드 2 및 노드 3에서 Tableau 관리 서비스를 중지합니다. 각 노드에서 다음 명령을 실행합니다.

   ```
   sudo /app/tableau_server/packages/scripts.<version_
   code>/./stop-administrative-services
   ```

3. PostgreSQL Step 3 tar를 복원합니다. Postgres를 실행하는 컴퓨터에서 다음 명령을 실행합니다.

   ```
   sudo su
   systemctl stop postgresql-12
   cd /var/lib/pgsql
   tar -xvf step3.12.bkp.tar
   systemctl start postgresql-12
   exit
   ```

4. 노드 1, 노드 2 및 노드 3에서 Tableau Step 3 tar를 복원합니다. 각 Tableau 노드에서 다음 명령을 실행합니다.

```
cd /data

sudo rm -rf tableau_data

sudo tar  -xvf step3.tableau_data.bkp.tar
```

5. Tableau 노드 1 컴퓨터에서 다음 파일을 제거합니다.

   - `sudo rm /data/tableau_data/data/tabsvc/appzookeeper/1/version-2/currentEpoch`
   - `sudo rm /data/tableau_data/data/tabsvc/appzookeeper/1/version-2/acceptedEpoch`
   - `sudo rm /data/tableau_data/data/tabsvc/tabadminagent/0/servicestate.json`

   셸에서 "파일을 찾을 수 없음" 오류가 반환되는 경우 파일 이름을 변경하여 경로의 `.../appzookeeper/<n>/version-2/...` 섹션에서 숫자 `<n>`을 늘려야 할 수 있습니다.

6. 노드 1, 노드 2 및 노드 3에서 관리 서비스를 다시 시작합니다. 각 노드에서 다음 명령을 실행합니다.

```
sudo /app/tableau_server/packages/scripts.<version_code>/./start-administrative-services

sudo su -l tableau -c "systemctl --user daemon-reload"

sudo /app/tableau_server/packages/scripts.<version_code>/./start-administrative-services
```

7. 노드 1에서 `tsm status` 명령을 실행하여 다시 시작 전에 TSM 상태를 모니터링합니다.

일부 경우 `Cannot connect to server...` 오류가 표시됩니다. 이 오류는 tabadmincontroller 서비스가 다시 시작되지 않았기 때문에 발생합니다. 계속해서 `tsm status`를 주기적으로 실행합니다. 10분 후에도 이 오류가 사라지지 않으면 `start-administrative-services` 명령을 다시 실행합니다.

몇 분 후에 `tsm status` 명령은 DEGRADED 상태를 반환하고 그런 다음 ERROR 상태를 반환합니다. STOPPED 상태가 반환될 때까지 TSM을 시작하지 마십시오. 그런 후 다음 명령을 실행합니다.

```
tsm start
```

노드 1~3에 조정 서비스를 배포하는 설치 프로세스를 다시 시작합니다.

노드 4 구성

노드 4의 구성 프로세스는 노드 3과 동일합니다.

위에 표시된 것과 동일한 명령 집합을 실행하되 명령에 `node4`를 지정(`node3` 대신)하여 노드 3에 설정한 것과 동일한 프로세스를 설정합니다.

노드 3을 확인할 때와 마찬가지로 `tsm status -v`를 실행하여 노드 4 구성을 확인합니다.

계속하기 전에 노드 4의 파일 저장소 프로세스 동기화가 완료될 때까지 기다립니다. 파일 저장소 서비스 상태는 완료될 때까지 `is synchronizing`을 반환합니다. 파일 저장소 서비스 상태가 `is running`을 반환하면 계속할 수 있습니다.

마지막 프로세스 구성 및 검증

구성 프로세스의 마지막 단계는 노드 1에서 중복 프로세스를 제거하는 것입니다.

1. 초기 노드(node1)에 연결합니다.

2. 노드 1의 파일 저장소를 해제합니다. 그러면 함께 배치된 컨트롤러에서 파일 저장소가 제거된다는 내용의 경고가 표시됩니다. 이 경고는 무시해도 됩니다. 다음 명령을 실행합니다.

   ```
   tsm topology filestore decommission -n node1
   ```

3. 파일 저장소가 해제되면 다음 명령을 실행하여 노드 1에서 백그라운더 프로세스를 제거합니다.

   ```
   tsm topology set-process -n node1 -pr backgrounder -c 0
   ```

4. 적용하기 전에 구성을 검토합니다. 다음 명령을 실행합니다.

   ```
   tsm pending-changes list
   ```

5. 변경 내용이 보류 목록에 있는지 확인한 후 변경 내용을 적용합니다.

   ```
   tsm pending-changes apply
   ```

   변경 내용을 적용하려면 다시 시작해야 합니다. 구성 및 다시 시작에는 시간이 조금 걸립니다.

6. 구성을 확인합니다.

   ```
   tsm status -v.
   ```

   계속하기 전에 노드 4의 파일 저장소 프로세스 동기화가 완료될 때까지 기다립니다. 파일 저장소 서비스 상태는 완료될 때까지 `is synchronizing`을 반환합니다. 파일 저장소 서비스 상태가 `is running`을 반환하면 계속할 수 있습니다.

백업 수행

Tableau Server의 전체 복구를 수행하려면 다음 세 가지 구성 요소가 포함된 백업 포트폴리오가 필요합니다.

- 리포지토리와 파일 저장소 데이터의 백업 파일. 이 파일은 `tsm maintenance backup` 명령에 의해 생성됩니다.
- 토폴로지 및 구성 내보내기 파일. 이 파일은 `tsm settings export` 명령에 의해 생성됩니다.
- 인증 인증서, 키 및 keytab 파일.

백업 및 복원 프로세스에 대한 자세한 설명은 Tableau Server 항목인 *Tableau Server의 전체 백업 및 복원 수행*(Linux | Linux)을 참조하십시오.

배포의 이 단계에서는 `tsm maintenance backup` 및 `tsm settings export` 명령을 실행하여 전체 복원에 필요한 모든 관련 파일 및 자산이 포함됩니다.

1. 다음 명령을 실행하여 구성 및 토폴로지 설정을 `ts_settings_backup.json`이라는 파일로 내보냅니다.

   ```
   tsm settings export -f ts_settings_backup.json
   ```

2. 다음 명령을 실행하여 이름이 `ts_backup-<yyyy-mm-dd>.tsbak`인 파일에 리포지토리 및 파일 저장소 데이터의 백업을 만듭니다. 파일 저장소가 컨트롤러 노드에 없다는 내용의 경고를 무시합니다.

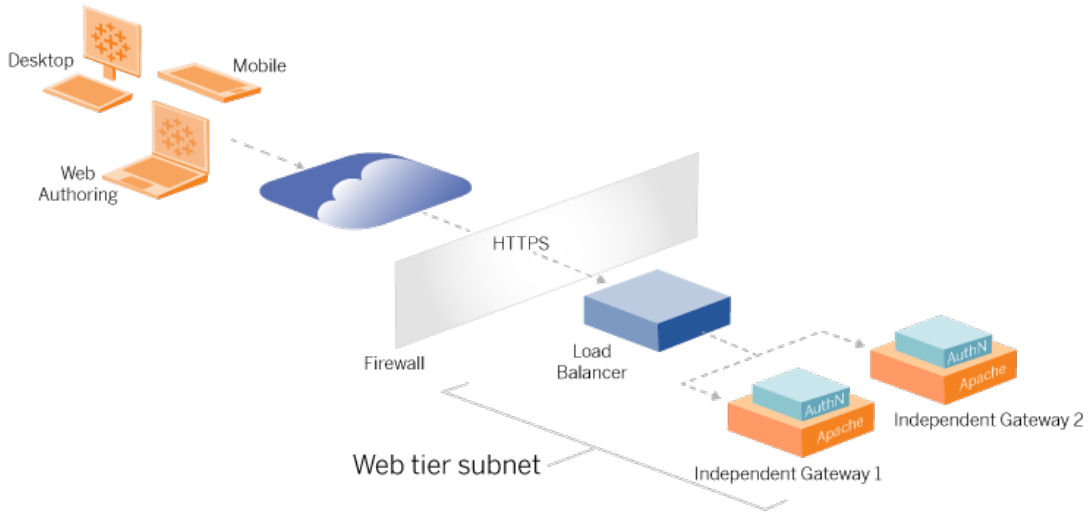
   ```
   tsm maintenance backup -f ts_backup -d --skip-compression
   ```

   백업 파일의 위치:

   ```
   /data/tableau_data/data/tabsvc/files/backups/
   ```

3. 두 파일을 모두 복사하고 Tableau Server 배포와 공유하지 않는 다른 저장소 자산에 저장합니다.

5부 - 웹 계층 구성



참조 아키텍처의 웹 계층에는 다음 구성 요소가 포함되어야 합니다.

- Tableau 클라이언트의 HTTPS 요청을 수락하고 역방향 프록시 서버로 전달하는 웹용 응용 프로그램 부하 분산 장치
- 역방향 프록시:
  - Tableau Server 독립 게이트웨이를 배포하는 것이 좋습니다.
  - 중복성을 구현하고 클라이언트 부하를 처리할 수 있도록 최소 2개의 프록시 서버를 구성하는 것이 좋습니다.
  - 부하 분산 장치의 HTTPS 트래픽을 수신합니다.
  - Tableau 호스트에 대한 고정 세션을 지원합니다.
  - 게이트웨이 프로세스를 실행하는 각 Tableau Server에 대한 라운드 로빈 부하 분산을 위한 프록시를 구성합니다.
  - 외부 IdP의 인증 요청을 처리합니다.
- 정방향 프록시: 라이선스 및 맵 기능을 위해 Tableau Server에서 인터넷에 액세스할 수 있어야 합니다. Tableau Service URL에 대한 정방향 프록시 허용 목록을 구성해야 합니다. *인터넷 통신*(Linux)을 참조하십시오.
- 모든 클라이언트 관련 트래픽은 HTTPS를 통해 암호화될 수 있습니다.
  - 클라이언트에서 응용 프로그램 부하 분산 장치로의 트래픽
  - 응용 프로그램 부하 분산 장치에서 역방향 프록시 서버로의 트래픽

- 프록시 서버에서 Tableau Server로의 트래픽
- 역방향 프록시에서 실행되는 인증 처리기에서 IdP로의 트래픽
- Tableau Server에서 IdP로의 트래픽

Tableau Server 독립 게이트웨이

Tableau Server 버전 2022.1에는 Tableau Server 독립 게이트웨이가 도입되었습니다. 독립 게이트웨이는 Tableau 게이트웨이 프로세스의 독립 실행형 인스턴스로, Tableau 인식 역방향 프록시 역할을 합니다.

독립 게이트웨이는 백엔드 Tableau Server로의 단순한 라운드 로빈 부하 분산을 지원합니다. 그러나 독립 게이트웨이는 엔터프라이즈 응용 프로그램 부하 분산 장치로 사용되도록 만들어진 것이 아닙니다. 따라서 독립 게이트웨이는 엔터프라이즈급 응용 프로그램 부하 분산 장치 뒤에서 실행하는 것이 좋습니다.

독립 게이트웨이에는 Advanced Management 라이선스가 필요합니다.

인증 및 권한 부여

기본 참조 아키텍처는 로컬 인증을 구성하여 Tableau Server를 설치할 것을 명시합니다. 이 모델에서 클라이언트는 Tableau Server에 연결하여 기본 Tableau Server의 로컬 인증 프로세스로 인증되어야 합니다. 참조 아키텍처에서는 이 인증 방법의 사용을 권장하지 않습니다. 이 시나리오에서는 인증되지 않은 클라이언트가 응용 프로그램 계층과 통신해야 하는데 이는 보안상 위험하기 때문입니다.

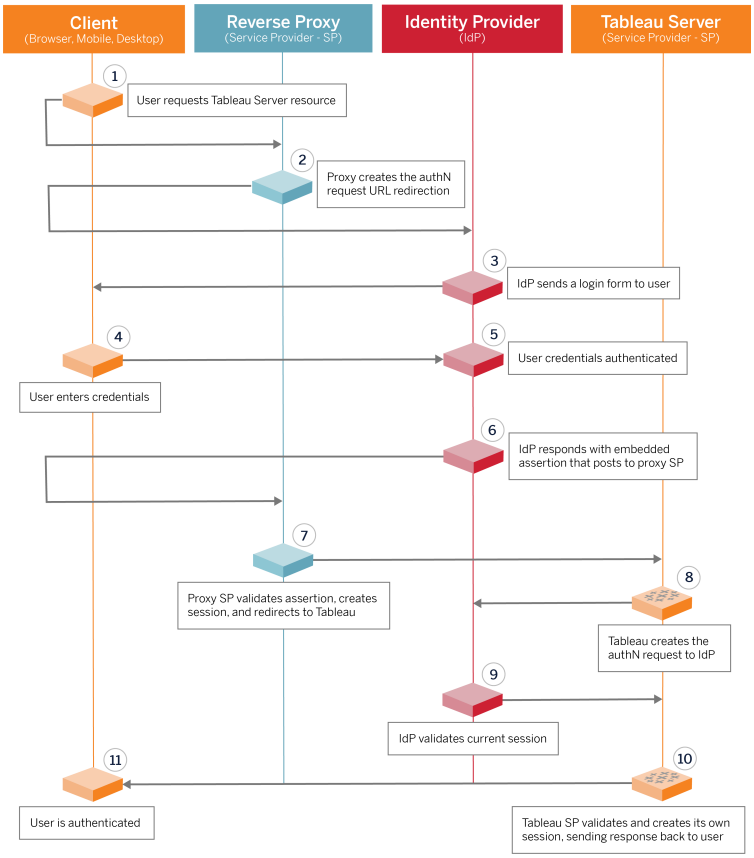
따라서 엔터프라이즈급 외부 ID 공급자를 구성하고 AuthN 모듈과 결합하여 응용 프로그램 계층으로 가는 모든 트래픽을 사전 인증하는 것이 좋습니다. 외부 IdP로 구성되면 기본 Tableau Server 로컬 인증 프로세스가 사용되지 않습니다. IdP가 사용자를 인증하면 Tableau Server가 배포의 리소스에 대한 액세스 권한을 부여합니다.

AuthN 모듈을 사용한 사전 인증

이 가이드에 문서화된 예제에는 SAML SSO가 구성되어 있지만 대부분의 외부 ID 공급자 및 AuthN 모듈을 사용하여 사전 인증 프로세스를 구성할 수 있습니다.

참조 아키텍처에서 역방향 프록시는 Tableau Server에 대한 요청을 프록시 연결하기 전에 IdP로 클라이언트 인증 세션을 만들도록 구성됩니다. 이 프로세스를 *사전 인증* 단계라고 합니다. 역방향 프록시는 인증된 클라이언트 세션만 Tableau Server로 리디렉션합니다. 그런 다음 Tableau Server에서 세션을 만들고 IdP로 세션 인증을 확인한 다음 클라이언트 요청을 반환합니다.

다음 다이어그램은 AuthN 모듈이 구성된 사전 인증 및 인증 프로세스에 대한 단계별 세부 정보를 보여줍니다. 일반 타사 솔루션 또는 Tableau Server 독립 게이트웨이를 역방향 프록시로 사용할 수 있습니다.



구성 개요

다음은 웹 계층을 구성하는 프로세스에 대한 개요입니다. 각 단계 후 연결을 확인합니다.

1. Tableau Server에 대한 HTTP 액세스를 제공하도록 역방향 프록시 2개를 구성합니다.
2. 프록시 서버에서 고정 세션을 사용하여 게이트웨이 프로세스를 실행하는 각 Tableau Server 인스턴스에 연결하는 부하 분산 논리를 구성합니다.
3. 인터넷 게이트웨이에서 고정 세션을 사용하여 요청을 역방향 프록시 서버로 전달하는 응용 프로그램 부하 분산을 구성합니다.
4. 외부 IdP를 통한 인증을 구성합니다. 역방향 프록시 서버에 인증 처리기를 설치하여 SSO 또는 SAML을 구성할 수 있습니다. AuthN 모듈은 외부 IdP와 Tableau 배포 간의 인증 핸드셰이크를 관리합니다. Tableau는 IdP 서비스 공급자 역할도 하며 IdP로 사용자를 인증합니다.
5. 이 배포에서 Tableau Desktop으로 인증하려면 클라이언트에서 Tableau Desktop 2021.2.1 이상을 실행해야 합니다.

Tableau Server 독립 게이트웨이를 사용한 웹 계층 구성 예제

이 항목의 나머지 부분에서는 예제 AWS 참조 아키텍처에서 Tableau Server 독립 게이트웨이를 사용하여 웹 계층을 구현하는 방법을 설명하는 전체 절차를 제공합니다. Apache를 역방향 프록시로 사용한 예제 구성은 부록 - Apache를 사용한 웹 계층 예제 배포를 참조하십시오.

예제 구성은 다음 구성 요소로 구성됩니다.

- AWS 응용 프로그램 부하 분산 장치
- Tableau Server 독립 게이트웨이
- Mellon 인증 모듈
- Okta IdP
- SAML 인증

> **참고**: 이 섹션에 나와 있는 웹 계층 구성 예제에는 타사 소프트웨어 및 서비스를 배포하는 자세한 절차가 포함되어 있습니다. Tableau는 웹 계층 시나리오를 지원하기 위한 절차를 확인하고 문서화하기 위해 최선을 다하고 있습니다. 하지만 타사 소프트웨어가 변경되거나 여기에 설명된 참조 아키텍처와 다른 시나리오가 있을 수 있습니다. 신뢰할 수 있는 구성 세부 정보 및 지원은 타사 설명서를 참조하십시오.

이 섹션 전반에 걸친 Linux 예는 RHEL와 같은 배포에 대한 명령을 보여줍니다. 특히 여기에 나온 명령은 Amazon Linux 2 배포를 통해 개발되었습니다. Ubuntu 배포를 실행 중인 경우 그에 따라 명령을 편집합니다.

이 예제의 웹 계층 배포는 단계별 구성 및 확인 절차를 따릅니다. 핵심 웹 계층 구성은 Tableau와 인터넷 간의 HTTP를 사용하도록 설정하는 다음 단계로 구성됩니다. AWS 응용 프로그램 부하 분산 장치 뒤에서 역방향 프록시/부하 분산을 수행하도록 독립 게이트웨이를 구성하고 실행합니다.

1. 환경 준비
2. 독립 게이트웨이 설치
3. 독립 게이트웨이 서버 구성
4. AWS 응용 프로그램 부하 분산 장치 구성

웹 계층을 설정하고 Tableau 연결을 확인한 후에는 외부 공급자를 통한 인증을 구성합니다.

환경 준비

독립 게이트웨이를 배포하기 전에 다음 작업을 완료합니다.

1. AWS 보안 그룹을 변경합니다. 사설 보안 그룹에서 오는 인바운드 독립 게이트웨이 하우스키핑 트래픽(TCP 21319)을 허용하도록 공용 보안 그룹을 구성합니다.

2. 4부 - Tableau Server 설치 및 구성에 문서화된 대로 버전 22.1.1 이상을 4노드 Tableau Server 클러스터에 설치합니다.

3. 호스트 컴퓨터 구성에 문서화된 대로 2개의 프록시 EC2 인스턴스를 공용 보안 그룹에 구성합니다.

독립 게이트웨이 설치

Tableau Server 독립 게이트웨이에는 Advanced Management 라이선스가 필요합니다.

Tableau Server 독립 게이트웨이 배포는 .rpm 패키지를 설치 후 실행한 다음 초기 상태를 구성하는 작업으로 구성됩니다. 이 가이드에 포함된 절차는 참조 아키텍처에 배포하기 위한 권장 지침을 제공합니다.

참조 아키텍처와 다른 배포의 경우 핵심 Tableau Server 설명서인 *독립 게이트웨이로 Tableau Server 설치*(Linux)를 참조하십시오.

> **중요**: 독립 게이트웨이를 구성하는 프로세스에서는 오류가 많이 발생합니다. 2개의 독립 게이트웨이 서버 인스턴스에서 구성 문제를 해결하기는 아주 어렵습니다. 따라서 한 번에 1개의 독립 게이트웨이 서버를 구성하는 것이 좋습니다. 첫 번째 서버를 구성한 후 기능을 확인하고 그 다음에 두 번째 독립 게이트웨이 서버를 구성해야 합니다.

각 독립 게이트웨이 서버를 따로 구성하게 되지만 공용 보안 그룹에 설치한 EC2 인스턴스 모두에서 이 설치 절차를 실행하십시오.

1. 업데이트를 실행하여 최신 수정을 Linux OS에 적용합니다.

   ```
   sudo yum update
   ```

2. Apache가 설치되어 있는 경우 제거합니다.

   ```
   sudo yum remove httpd
   ```

3. Tableau 다운로드 페이지에서 버전 2022.1.1 이상 독립 게이트웨이 설치 패키지를 Tableau Server를 실행할 호스트 컴퓨터에 복사합니다.

예를 들어 Linux RHEL 유사 운영 체제를 실행하는 컴퓨터에서 다음을 실행합니다.

```
wget
https://downloads.tableau.com/esdalt/2022<version>/tableau-
server-tsig-<version>.x86_64.rpm
```

4. 설치 프로그램을 실행합니다. 예를 들어 Linux RHEL과 같은 운영 체제에서 다음을 실행합니다.

```
sudo yum install <tableau-tsig-version>.x86_64.rpm
```

5. `/opt/tableau/tableau_tsig/packages/scripts.<version_code>/` 디렉터리로 변경하고 거기에 있는 `initialize-tsig` 스크립트를 실행합니다. `--accepteula` 플래그에 더해 Tableau Server 배포가 실행되는 서브넷의 IP 범위를 제공해야 합니다. IP 범위를 지정하려면 `-c` 옵션을 사용합니다. 아래의 예제는 지정된 예제 AWS 서브넷과 명령을 보여줍니다.

```
sudo ./initialize-tsig --accepteula -c "ip 10.0.30.0/24
10.0.31.0/24"
```

6. 초기화가 완료되면 `tsighk-auth.conf` 파일을 열고 파일의 인증 암호를 복사합니다. 백엔드 Tableau Server 구성의 일부로 이 코드를 각 독립 게이트웨이 인스턴스에 대해 제출해야 합니다.

```
sudo less /var/opt/tableau/tableau_tsig/config/tsighk-auth.conf
```

7. 독립 게이트웨이의 두 인스턴스에서 이전 단계를 실행한 후 `tsig.json` 구성 파일을 준비합니다. 구성 파일은 “independentGateways” 배열로 구성됩니다. 이 배열에는 각각 독립 게이트웨이 인스턴스의 연결 세부 정보를 정의하는 구성 개체가 포함됩니다.

   다음 JSON을 복사하고 배포 환경에 따라 사용자 지정합니다. 여기의 예제는 샘플 AWS 참조 아키텍처에 대한 파일을 보여줍니다.

아래의 예제 JSON 파일에는 독립 게이트웨이 1개의 연결 정보만 포함되어 있습니다. 이 프로세스의 후반부에서 두 번째 독립 게이트웨이 서버의 연결 정보를 포함할 것입니다.

다음 절차를 위해 파일을 `tsig.json`으로 저장합니다.

```
{
"independentGateways": [
 {
       "id": "ip-10-0-1-169.ec2.internal",
       "host": "ip-10-0-1-169.ec2.internal",
       "port": "21319",
       "protocol" : "http",
       "authsecret": "13660-27118-29070-25482-9518-22453"
       }]
 }
```

- `"id"` - 독립 게이트웨이를 실행하는 AWS EC2 인스턴스의 사설 DNS 이름입니다.
- `"host"` - `"id"`와 동일합니다.
- `"port"` - 하우스키핑 포트이며 기본값은 `"21319"`입니다.
- `"protocol"` - 클라이언트 트래픽의 프로토콜입니다. 초기 구성의 경우 `http`로 남겨둡니다.
- `"authsecret"` - 이전 단계에서 복사한 암호입니다.

독립 게이트웨이: 직접 연결과 릴레이 연결

계속하기 전에 배포에서 구성할 연결 체계(직접 연결 또는 릴레이 연결)를 결정해야 합니다. 여기에 관련된 의사 결정 데이터 요소와 함께 각 옵션이 간략하게 설명되어 있습니다.

릴레이 연결: 단일 포트를 통해 Tableau Server의 게이트웨이 프로세스로 클라이언트 통신을 릴레이하도록 독립 게이트웨이를 구성할 수 있습니다. 이를 *릴레이* 연결이라고 합니다.

- 릴레이 프로세스에서는 독립 게이트웨이에서 백엔드 Tableau Server 게이트웨이 프로세스에 연결할 때 추가 홉이 발생합니다. 이 추가 홉으로 인해 직접 연결 구성에 비해 성능이 저하됩니다.
- 릴레이 모드에는 TLS가 지원됩니다. 릴레이 모드의 모든 통신은 단일 프로토콜(HTTP 또는 HTTPS)로 제한되므로 TLS를 통해 암호화하고 인증할 수 있습니다.

직접 연결: 독립 게이트웨이는 여러 포트를 통해 직접 백엔드 Tableau Server 프로세스와 통신할 수 있습니다. 이 통신 방식을 *직접* 연결이라고 합니다.

- 백엔드 Tableau Server에 직접 연결되기 때문에 릴레이 연결에 비해 클라이언트 성능이 크게 개선됩니다.
- 독립 게이트웨이에서 Tableau Server 컴퓨터로의 직접 프로세스 통신에는 공용에서 사설로 연결되는 서브넷에서 16개 이상의 포트를 열어야 합니다.
- 독립 게이트웨이에서 Tableau Server로의 프로세스에 TLS는 아직 지원되지 않습니다.

릴레이 연결 구성

Tableau Server와 독립 게이트웨이 간에 TLS를 실행하려면 릴레이 연결로 구성해야 합니다. EDG의 예제 시나리오는 릴레이 연결로 구성됩니다.

1. `tsig.json`을 Tableau Server 배포의 노드 1에 복사합니다.

2. 노드 1에서 다음 명령을 실행하여 독립 게이트웨이를 사용하도록 설정합니다.

```
tsm stop
tsm configuration set -k gateway.tsig.proxy_tls_optional -v
none
tsm pending-changes apply
tsm topology external-services gateway enable -c tsig.json
tsm start
```

직접 연결 구성

직접 연결은 TLS를 지원하지 않으므로 다른 방법으로 모든 네트워크 트래픽을 보호할 수 있는 경우에만 직접 연결을 구성하는 것이 좋습니다. Tableau Server와 독립 게이트웨이 간에 TLS를 실행하려면 릴레이 연결로 구성해야 합니다. EDG의 예제 시나리오는 릴레이 연결로 구성됩니다.

Tableau Server에 직접 연결하도록 독립 게이트웨이를 구성하는 경우 통신을 트리거하는 구성을 사용해야 합니다. Tableau Server가 독립 게이트웨이와 통신하면 프로토콜 대상이 설정됩니다. 그러면 독립 게이트웨이 컴퓨터에서 `proxy_targets.csv`를 검색하고 AWS의 공용에서 사설 보안 그룹으로 연결되는 해당하는 포트를 열어야 합니다.

1. `tsig.json`을 Tableau Server 배포의 노드 1에 복사합니다.
2. 노드 1에서 다음 명령을 실행하여 독립 게이트웨이를 사용하도록 설정합니다.

   ```
   tsm stop
   tsm topology external-services gateway enable -c tsig.json
   tsm start
   ```

3. 독립 게이트웨이 컴퓨터에서 다음 명령을 실행하여 Tableau Server 클러스터에 사용되고 있는 포트를 확인합니다.

   ```
   less /var/opt/tableau/tableau_tsig/config/httpd/proxy_
   targets.csv
   ```

4. AWS 보안 그룹을 구성합니다. `proxy_targets.csv`에 나열된 TCP 포트를 추가하여 공용 보안 그룹과 사설 보안 그룹 간의 통신을 허용합니다.

   Tableau Server 배포가 변경될 경우 포트가 변경될 수 있으므로 포트 수신 구성을 자동화하는 것이 좋습니다. Tableau Server 배포에서 노드를 추가하거나 프로세스를 재구성하면 독립 게이트웨이에 필요한 포트 액세스 권한이 변경됩니다.

확인: 기본 토폴로지 구성

`http://<gateway-public-IP-address>`로 이동하여 Tableau Server 관리 페이지에 액세스할 수 있어야 합니다.

Tableau Server 로그인 페이지가 로드되지 않거나 Tableau Server가 시작되지 않으면 다음 문제 해결 단계를 따르십시오.

네트워크:

- Tableau Server 노드 1에서 `wget` 명령(wget http://<internal IP address of Independent Gateway>:21319)을 실행하여 Tableau 배포와 독립 게이트웨이 인스턴스 간의 연결을 확인합니다. 예를 들면 다음과 같습니다.

  ```
  wget http://ip-10-0-1-38:21319
  ```

  연결이 거부되거나 실패하면 공용 보안 그룹이 사설 보안 그룹의 독립 게이트웨이 하우스키핑 트래픽(TCP 21319)을 허용하도록 구성되었는지 확인합니다.

  보안 그룹이 올바르게 구성된 경우 독립 게이트웨이를 초기화하는 동안 올바른 IP 주소 또는 IP 범위를 지정했는지 확인합니다. `/etc/opt/tableau/tableau_tsig/environment.bash`에 있는 `environment.bash` 파일에서 이 구성을 보고 변경할 수 있습니다. 이 파일을 변경한 경우 아래 설명된 대로 tsig-http 서비스를 다시 시작하십시오.

프록시 1 호스트:

1. httpd.conf 파일을 독립 게이트웨이 stub 파일로 덮어씁니다.

   ```
   cp /var/opt/tableau/tableau_tsig/config/httpd.conf.stub
   /var/opt/tableau/tableau_tsig/config/httpd.conf
   ```

2. 첫 번째 문제 해결 단계로 tsig-httpd를 다시 시작합니다.

   ```
   sudo su - tableau-tsig
   systemctl --user restart tsig-httpd
   exit
   ```

Tableau 노드 1:

- `tsig.json` 파일을 다시 확인합니다. 오류가 발견되면 수정한 다음 `tsm topology external-services gateway update -c tsig.json`을 실행합니다.
- 직접 연결을 실행하는 경우 `proxy_targets.csv`에 나열된 TCP 포트가 공용 보안 그룹과 사설 보안 그룹 간의 수신 포트로 구성되었는지 확인합니다.

AWS 응용 프로그램 부하 분산 장치 구성

부하 분산 장치를 HTTP 수신기로 구성합니다. 다음 절차에서는 AWS에서 부하 분산 장치를 추가하는 방법에 대해 설명합니다.

1단계: 대상 그룹 만들기

대상 그룹은 프록시 서버를 실행하는 EC2 인스턴스를 정의하는 AWS 구성입니다. 이러한 그룹은 LBS에서 오는 트래픽의 대상입니다.

1. EC2 > **대상 그룹** > **대상 그룹 만들기**
2. 만들기 페이지에서:
   - 대상 그룹 이름 입력(예: `TG-internal-HTTP`)
   - 대상 유형: 인스턴스
   - 프로토콜: HTTP
   - 포트: 80
   - VPC: 해당하는 VPC 선택
   - **상태 확인** > **고급 상태 확인 설정** > **성공 코드**에서 읽을 코드 목록(`200,303`)을 추가합니다.
   - **만들기**를 클릭합니다.
3. 방금 만든 대상 그룹을 선택한 다음 **대상** 탭을 클릭합니다.
   - **편집**을 클릭합니다.
   - 프록시 애플리케이션을 실행하는 EC2 인스턴스(또는 한 번에 1개를 구성하는 경우 단일 인스턴스)를 선택한 다음 **등록된 항목에 추가**를 클릭합니다.
   - **저장**을 클릭합니다.

2단계: 부하 분산 장치 마법사 시작

1. EC2 > **부하 분산 장치** > **부하 분산 장치 만들기**
2. "부하 분산 장치 유형 선택" 페이지에서 응용 프로그램 부하 분산 장치를 만듭니다.

> **참고**: 부하 분산 장치를 구성하기 위해 표시되는 UI는 AWS 데이터 센터 간에 일관되지 않습니다. 아래의 "마법사 구성" 절차는 **1단계 부하 분산 장치 구성**으로 시작하는 AWS 구성 마법사를 설명합니다.
>
> 데이터 센터가 모든 구성을 단일 페이지에 표시하는 경우 페이지 아래쪽에 **부하 분산 장치 만들기** 단추가 포함되어 있으면 아래의 "단일 페이지 구성" 절차를 따르십시오.

마법사 구성

1. **부하 분산 장치 구성** 페이지:
   - 이름 지정
   - 스키마: 인터넷 연결(기본값)
   - IP 주소 유형: ipv4(기본값)
   - 수신기(수신기 및 라우팅):
     a. 기본 HTTP 수신기를 그대로 둡니다.
     b. **수신기 추가**를 클릭하고 `HTTPS:443`를 추가합니다.
   - VPC: 모든 항목을 설치한 VPC 선택
   - 가용성 영역:
     - 데이터 센터 지역을 나타내는 **a** 및 **b**를 선택합니다.
     - 해당하는 각 드롭다운 선택 도구에서 공용 서브넷(프록시 서버가 상주하는 서브넷)을 선택합니다.
   - **보안 설정 구성** 클릭
2. **보안 설정 구성** 페이지

- 공용 SSL 인증서를 업로드합니다.
- **다음: 보안 그룹 구성**을 클릭합니다.

3. **보안 그룹 구성** 페이지:
   - 공용 보안 그룹을 선택합니다. 기본 보안 그룹이 선택된 경우 해당 선택을 취소합니다.
   - **다음: 라우팅 구성**을 클릭합니다.
4. **라우팅 구성** 페이지
   - 대상 그룹: 기존 대상 그룹
   - 이름: 이전에 만든 대상 그룹 선택
   - **다음: 대상 등록**을 클릭합니다.
5. **대상 등록** 페이지
   - 이전에 구성한 프록시 서버 인스턴스 2개가 표시됩니다.
   - **다음: 검토**를 클릭합니다.
6. **검토** 페이지

   만들기를 클릭합니다.

단일 페이지 구성

기본 구성

- 이름 지정
- 스키마: 인터넷 연결(기본값)
- IP 주소 유형: ipv4(기본값)

네트워크 매핑

- VPC: 모든 항목을 설치한 VPC 선택
- 매핑:
  - 데이터 센터 지역을 나타내는 **a** 및 **b**(또는 그에 상응하는) 가용성 영역을 선택합니다.

- 해당하는 각 드롭다운 선택 도구에서 공용 서브넷(프록시 서버가 상주하는 서브넷)을 선택합니다.

보안 그룹

공용 보안 그룹을 선택합니다. 기본 보안 그룹이 선택된 경우 해당 선택을 취소합니다.

수신기 및 라우팅

- 기본 HTTP 수신기를 그대로 둡니다. **기본 동작**의 경우 이전에 설정한 대상 그룹을 지정합니다.
- **수신기 추가**를 클릭하고 `HTTPS:443`을 추가합니다. **기본 동작**의 경우 이전에 설정한 대상 그룹을 지정합니다.

보안 수신기 설정

- 공용 SSL 인증서를 업로드합니다.

부하 분산 장치 만들기를 클릭합니다.

3단계: 연결 유지 사용

1. 부하 분산 장치를 만든 후 대상 그룹에서 연결 유지를 사용하도록 설정해야 합니다.
    - AWS 대상 그룹 페이지(**EC2 > 부하 분산 > 대상 그룹**)를 열고 방금 설정한 대상 그룹 인스턴스를 선택합니다. **동작** 메뉴에서 **특성 편집**을 선택합니다.
    - **특성 편집** 페이지에서 **연결 유지**를 선택하고 기간을 `1 day`로 지정한 다음 **변경 내용을 저장**합니다.
2. 부하 분산 장치에서 HTTP 수신기에 연결 유지를 사용하도록 설정합니다. 방금 구성한 부하 분산 장치를 선택한 다음 **수신기** 탭을 클릭합니다.
    - **HTTP:80**에서 **규칙 보기/편집**을 클릭합니다. 결과로 표시되는 **규칙** 페이지에서 편집 아이콘을 클릭하여 규칙을 편집합니다(페이지 맨 위에서 한 번 클릭하여 편집한 다음 규칙별로 다시 편집). 기존 THEN 규칙을 삭제하고 **동작 추가 > 전달 대상...**을 클릭하여 바꿉니다. 결과로 나온 THEN 구성에서 이전에 만든 것과 동일한 대상 그룹을 지정합니다. 그룹 수준 연결 유지에서 연결 유지를 사용하도록 설정하고 기간을 1일로 설정합니다. 설정을 저장한 다음 **업데이트**를 클릭합니다.

4단계: 부하 분산 장치에서 유휴 시간 초과 설정

부하 분산 장치에서 유휴 시간 초과를 400초로 업데이트합니다.

이 배포에 대해 구성한 부하 분산 장치를 선택한 다음 **동작 > 특성 편집**을 클릭합니다. **유휴 시간 초과**를 400 초로 설정한 다음 **저장**을 클릭합니다.

5단계: LBS 연결 확인

AWS 부하 분산 장치 페이지(**EC2 > 부하 분산 장치**)를 열고 방금 설정한 부하 분산 장치 인스턴스를 선택합니다.

설명에서 DNS 이름을 복사하여 브라우저에 붙여 넣고 Tableau Server 로그인 페이지에 액세스합니다.

500 수준 오류가 발생하면 프록시 서버를 다시 시작해야 할 수 있습니다.

공용 Tableau URL로 DNS 업데이트

AWS 부하 분산 장치 설명의 도메인 DNS 영역 이름을 사용하여 DNS에 CNAME 값을 만듭니다. URL(tableau.example.com)에 대한 트래픽은 AWS 공용 DNS 이름으로 전송되어야 합니다.

연결 확인

DNS 업데이트가 완료되면 공용 URL(예: `https://tableau.example.com`)을 입력하여 Tableau Server 로그인 페이지로 이동할 수 있어야 합니다.

인증 구성 예: SAML 및 외부 IdP

다음 예에서는 AWS 참조 아키텍처에서 실행되는 Tableau 배포를 위해 Okta IdP 및 Mellon 인증 모듈로 SAML을 설정하고 구성하는 방법에 대해 설명합니다.

이 예제는 이전 섹션의 정보를 사용하며 한 번에 1개의 독립 게이트웨이 서버를 구성한다고 가정합니다.

이 예에서는 HTTP를 통해 Tableau Server와 독립 게이트웨이를 구성하는 방법을 설명합니다. Okta는 HTTPS를 통해 AWS 부하 분산 장치로 요청을 보내지만 모든 내부 트래픽은 HTTP를 통해 이동합니다. 이 시나리오에서 구성할 때는 URL 문자열을 설정할 때 HTTP와 HTTPS 프로토콜의 차이점을 숙지하십시오.

이 예에서는 Mellon을 독립 게이트웨이 서버의 사전 인증 서비스 공급자 모듈로 사용합니다. 이 구성에서는 인증된 트래픽만 Tableau Server에 연결되며 Tableau Server는 Okta IdP를 통한 서비스 공급자 역할도 합니다. 따라서 Mellon 서비스 공급자용과 Tableau 서비스 공급자용으로 IdP 응용 프로그램 2개를 구성해야 합니다.

Tableau 관리자 계정 만들기

SAML을 구성할 때 흔히 하는 실수는 SSO를 사용하도록 설정하기 전에 Tableau Server에서 관리자 계정을 생성하는 것을 잊는 것입니다.

첫 번째 단계는 Tableau Server에서 서버 관리자 역할의 계정을 만드는 것입니다. Okta 시나리오의 예에서 사용자 이름은 올바른 이메일 주소 형식이어야 합니다(예: user@example.com). 이 사용자에 대한 비밀번호를 설정해야 하지만 SAML이 구성된 후에는 비밀번호가 사용되지 않습니다.

Okta 사전 인증 응용 프로그램 구성

이 섹션에 설명된 전체 시나리오에서는 두 개의 Okta 응용 프로그램이 필요합니다.

- Okta 사전 인증 응용 프로그램
- Okta Tableau Server 응용 프로그램

이러한 각 응용 프로그램은 서로 다른 메타데이터에 연결되며 이를 역방향 프록시와 Tableau Server에서 각각 구성해야 합니다.

이 절차에서는 Okta 사전 인증 응용 프로그램을 만들고 구성하는 방법에 대해 설명합니다. 이 항목의 뒷부분에서는 Okta Tableau Server 응용 프로그램을 만듭니다. 사용자 수가 제한되어 있는 Okta 무료 테스트 계정은 Okta 개발자 웹 페이지를 참조하십시오.

Mellon 사전 인증 서비스 공급자를 위한 SAML 앱 통합을 만듭니다.

1. Okta 관리 대시보드 > **응용 프로그램** > **앱 통합 만들기**를 엽니다.

2. **새 앱 통합 만들기** 페이지에서 **SAML 2.0**을 선택한 후 **다음**을 클릭합니다.

3. **일반 설정** 탭에서 앱 이름(예: `Tableau Pre-Auth`)을 입력하고 **다음**을 클릭합니다.

4. **SAML 구성** 탭에서:

   - SSO(Single Sign-On) URL. Single Sign-On URL의 최종 경로 요소는 이 절차의 뒷부분에서 설명하는 `mellon.conf` 구성 파일의 `MellonEndpointPath`를 일컫습니다. 원하는 끝점을 지정할 수 있습니다. 이 예에서는 `sso`가 끝점입니다. 마지막 요소인 `postResponse`는 `https://tableau.example.com/sso/postResponse`에 필수입니다.
   - **수신자 URL 및 대상 URL에 사용** 확인란을 선택 취소합니다.
   - 수신자 URL: SSO URL과 동일하지만 HTTP가 포함됩니다. 예를 들어 `http://tableau.example.com/sso/postResponse`입니다.
   - 대상 URL: SSO URL과 동일하지만 HTTP가 포함됩니다. 예를 들어 `http://tableau.example.com/sso/postResponse`입니다.
   - 대상 URI(SP 엔터티 ID) 예를 들어 `https://tableau.example.com`입니다.
   - 이름 ID 형식: `EmailAddress`
   - 응용 프로그램 사용자 이름: `Email`
   - 특성 문: 이름 = `mail`, 이름 형식 = `Unspecified`, 값 = `user.email`.

   다음을 클릭합니다.

5. **피드백** 탭에서 다음을 선택합니다.

   - **내부 앱을 추가하는 Okta 고객**
   - **이전에 만든 내부 앱**
   - **마침**을 클릭합니다.

6. 사전 인증 IdP 메타데이터 파일을 만듭니다.

   - Okta에서: **Applications(응용 프로그램)** > **Applications(응용 프로그램)** > Your new application(새 응용 프로그램)(예: `Tableau Pre-Auth`) > **Sign On (로그온)**

- **SAML Signing Certificates(SAML 서명 인증서)** 근처에서 **View SAML setup instructions(SAML 설정 지침 보기)**를 클릭합니다.
- **How to Configure SAML 2.0 for <pre-auth> Application(<pre-auth> 응용 프로그램에 대해 SAML 2.0을 구성하는 방법)** 페이지에서 **Optional(선택 사항)** 섹션인 **Provide the following IDP metadata to your SP provider(SP 공급자에게 다음 IdP 메타데이터 제공)**로 스크롤합니다.
- XML 필드의 콘텐츠를 복사하고 `pre-auth_idp_metadata.xml`이라는 파일에 저장합니다.

7. (선택 사항) 다단계 인증을 구성합니다.

   - Okta에서: **Applications(응용 프로그램)** > **Applications(응용 프로그램)** > Your new application(새 응용 프로그램)(예: `Tableau Pre-Auth`) > **Sign On (로그온)**
   - **로그온 정책**에서 **규칙 추가**를 클릭합니다.
   - **앱 로그온 규칙**에서 이름 및 다른 MFA 옵션을 지정합니다. 기능을 테스트하려면 모든 옵션을 기본값으로 두어도 됩니다. 그러나 **동작**에서는 **인증 요소 표시**를 선택하고 사용자가 로그인해야 하는 빈도를 지정해야 합니다. **저장**을 클릭합니다.

Okta 사용자 만들기 및 할당

1. Okta에서 **디렉터리** > **사용자** > **사용자 추가**로 이동하여 Tableau에서 만든 것과 동일한 사용자 이름(user@example.com)으로 사용자를 만듭니다.
2. 사용자가 만들어지면 해당 사용자에게 새 Okta 앱을 할당합니다. 사용자 이름을 클릭한 다음 **응용 프로그램 할당**에서 응용 프로그램을 할당합니다.

사전 인증을 위해 Mellon 설치

이 예제에서는 주요 오픈 소스 모듈인 mod_auth_mellon을 사용합니다. 일부 Linux 배포는 이전 리포지토리의 사용되지 않는 mod_auth_mellon 버전을 패키지합니다. 이 사용되지 않는 버전에는 알 수 없는 보안 취약점이나 기능 문제가 포함될 수 있습니다. mod_auth_mellon을 사용하도록 선택하는 경우 최신 버전을 사용 중인지 확인하십시오.

mod_auth_mellon 모듈은 타사 소프트웨어입니다. Tableau는 이 시나리오를 지원하기 위한 절차를 확인하고 문서화하기 위해 최선을 다하고 있습니다. 하지만 타사 소프트웨어

가 변경되거나 여기에 설명된 참조 아키텍처와 다른 시나리오가 있을 수 있습니다. 신뢰할 수 있는 구성 세부 정보 및 지원은 타사 설명서를 참조하십시오.

1. 독립 게이트웨이를 실행하는 활성 EC2 인스턴스에서 최신 버전의 Mellon 인증 모듈을 설치합니다.

2. `/etc/mellon` 디렉터리를 만듭니다.

   ```
   sudo mkdir /etc/mellon
   ```

Mellon을 사전 인증 모듈로 구성

독립 게이트웨이를 처음 설치하는 경우 다음 절차를 실행합니다.

Okta 구성에서 만든 `pre-auth_idp_metadata.xml` 파일의 복사본이 있어야 합니다.

1. 디렉터리를 변경합니다.

   ```
   cd /etc/mellon
   ```

2. 서비스 공급자 메타데이터를 만듭니다. `mellon_create_metadata.sh` 스크립트를 실행합니다. 명령에 조직의 엔티티 ID와 반환 URL을 포함해야 합니다.

   반환 URL은 Okta의 *Single Sign On URL*이라고 합니다. 반환 URL의 최종 경로 요소는 이 절차의 뒷부분에서 설명하는 `mellon.conf` 구성 파일의 `MellonEndpointPath`를 일컫습니다. 이 예에서는 끝점 경로로 `sso`를 지정합니다.

   예를 들면 다음과 같습니다.

   ```
   sudo /usr/libexec/mod_auth_mellon/mellon_create_metadata.sh
   https://tableau.example.com "https://tableau.example.com/sso"
   ```

   스크립트는 서비스 공급자 인증서, 키 및 메타데이터 파일을 반환합니다.

3. 쉽게 읽을 수 있도록 `mellon` 디렉터리에서 서비스 공급자 파일의 이름을 바꿉니다. 설명서에서 이러한 파일은 다음 이름으로 설명하겠습니다.

```
sudo mv *.key mellon.key
sudo mv *.cert mellon.cert
sudo mv *.xml sp_metadata.xml
```

4. pre-auth_idp_metadata.xml 파일을 동일한 디렉터리에 복사합니다.

5. /etc/mellon 디렉터리의 모든 파일에 대한 소유권과 사용 권한을 변경합니다.

```
sudo chown tableau-tsig mellon.key
sudo chown tableau-tsig mellon.cert
sudo chown tableau-tsig sp_metadata.xml
sudo chown tableau-tsig pre-auth_idp_metadata.xml
sudo chmod +r * mellon.key
sudo chmod +r * mellon.cert
sudo chmod +r * sp_metadata.xml
sudo chmod +r * pre-auth_idp_metadata.xml
```

6. /etc/mellon/conf.d 디렉터리를 만듭니다.

```
sudo mkdir /etc/mellon/conf.d
```

7. /etc/mellon/conf.d 디렉터리에서 global.conf 파일을 만듭니다.

   아래와 같이 파일 내용을 복사하고 루트 도메인 이름으로 MellonCookieDomain 을 업데이트합니다. 예를 들어 Tableau의 도메인 이름이 tableau.example.com 인 경우 루트 도메인에 대해 example.com을 입력합니다.

```
<Location "/">
AuthType Mellon
MellonEnable auth
Require valid-user
MellonCookieDomain <root domain>
MellonSPPrivateKeyFile /etc/mellon/mellon.key
MellonSPCertFile /etc/mellon/mellon.cert
MellonSPMetadataFile /etc/mellon/sp_metadata.xml
```

```
MellonIdPMetadataFile /etc/mellon/pre-auth_idp_metadata.xml
MellonEndpointPath /sso
</Location>

<Location "/tsighk">
MellonEnable Off
</Location>
```

8. `/etc/mellon/conf.d` 디렉터리에서 `mellonmod.conf` 파일을 만듭니다.

   이 파일에는 `mod_auth_mellon.so` 파일의 위치를 지정하는 단일 지시문이 포함됩니다. 여기 예제에서 위치는 파일의 기본 위치입니다. 파일이 이 위치에 있는지 확인하거나 이 지시문의 경로를 `mod_auth_mellon.so`의 실제 위치와 일치하도록 변경합니다.

   ```
   LoadModule auth_mellon_module /usr/lib64/httpd/modules/mod_
   auth_mellon.so
   ```

Okta에서 Tableau Server 응용 프로그램 만들기

1. Okta 대시보드에서: **응용 프로그램 > 응용 프로그램 > 앱 카탈로그 찾아보기**
2. **앱 통합 카탈로그 찾아보기**에서 `Tableau`를 검색하고 Tableau Server 타일을 선택한 다음 **추가**를 클릭합니다.
3. **Tableau Server 추가 > 일반 설정**에서 레이블을 입력하고 **다음**을 클릭합니다.
4. 로그온 옵션에서 **SAML 2.0**을 선택한 다음 고급 로그온 설정으로 스크롤합니다.
   - **SAML 엔터티 ID**: 공용 URL을 입력합니다(예: https://tableau.example.com).
   - **응용 프로그램 사용자 이름 형식**: 이메일
5. **ID 공급자 메타데이터** 링크를 클릭하여 브라우저를 시작합니다. 브라우저 링크를 복사합니다. 이 링크는 다음 절차에서 Tableau를 구성할 때 사용됩니다.
6. **완료**를 클릭합니다.
7. 사용자(user@example.com)에게 새 Tableau Server Okta 앱 할당: 사용자 이름을 클릭하고 **응용 프로그램 할당**에서 응용 프로그램을 할당합니다.

Tableau Server에서 인증 모듈 구성 설정

Tableau Server 노드 1에서 다음 명령을 실행합니다. 이러한 명령은 원격 독립 게이트웨이 컴퓨터의 Mellon 구성 파일에 대한 파일 위치를 지정합니다. 이러한 명령에 지정된

파일 경로가 원격 독립 게이트웨이 컴퓨터의 경로 및 파일 위치에 매핑되는지 다시 확인하십시오.

```
tsm configuration set -k gateway.tsig.authn_module_block -v
"/etc/mellon/conf.d/mellonmod.conf" --force-keys
tsm configuration set -k gateway.tsig.authn_global_block -v
"/etc/mellon/conf.d/global.conf" --force-keys
```

가동 중단을 줄이려면 다음 섹션에 설명된 대로 SAML을 설정하기 전까지 변경 내용을 적용하지 마십시오.

Tableau Server에서 IdP에 SAML을 사용하도록 설정

Tableau Server 노드 1에서 이 절차를 실행합니다.

1. Okta에서 Tableau Server 응용 프로그램 메타데이터를 다운로드합니다. 이전 절차에서 저장한 링크를 사용합니다.

   ```
   wget https://dev-
   66144217.okta.com/app/exk1egxgt1fhjkSeS5d7/sso/saml/metadata -O
   idp_metadata.xml
   ```

2. TLS 인증서와 관련 키 파일을 Tableau Server에 복사합니다. 키 파일은 RSA 키여야 합니다. SAML 인증서 및 IdP 요구 사항에 대한 자세한 내용은 *SAML 요구 사항* (Linux)을 참조하십시오.

   인증서 관리 및 배포를 간소화하고 보안을 위한 모범 사례를 적용하기 위해 신뢰할 수 있는 주요 외부 CA(인증 기관)에서 생성된 인증서를 사용할 것을 권장합니다. 또는 자체 서명 인증서를 생성하거나 TLS용 PKI의 인증서를 사용할 수도 있습니다.

   TLS 인증서가 없는 경우 아래 포함된 절차를 사용하여 자체 서명 인증서를 생성할 수 있습니다.

자체 서명 인증서를 생성합니다.

Tableau Server 노드 1에서 이 절차를 실행합니다.

a. 서명 루트 CA(인증 기관)키를 생성합니다.

```
openssl genrsa -out rootCAKey-saml.pem 2048
```

b. 루트 CA 인증서를 만듭니다.

```
openssl req -x509 -sha256 -new -nodes -key rootCAKey-
saml.pem -days 3650 -out rootCACert-saml.pem
```

인증서 필드에 값을 입력하라는 메시지가 표시됩니다. 예:

```
Country Name (2 letter code) [XX]:US
State or Province Name (full name) []:Washington
Locality Name (eg, city) [Default City]:Seattle
Organization Name (eg, company) [Default Company
Ltd]:Tableau
Organizational Unit Name (eg, section) []:Operations
Common Name (eg, your name or your server's hostname)
[]:tableau.example.com
Email Address []:example@tableau.com
```

c. 인증서 및 관련 키(아래 예에서 `server-saml.csr` 및 `server-saml.key`)를 만듭니다. 인증서의 주체 이름은 Tableau 호스트의 공용 호스트 이름과 일치해야 합니다. 주체 이름은 `-subj` 옵션과 `"/CN=<host-name>"` 형식을 사용하여 설정됩니다. 예:

```
openssl req -new -nodes -text -out server-saml.csr -keyout
server-saml.key -subj "/CN=tableau.example.com"
```

d. 위에서 만든 CA 인증서를 사용하여 새 인증서에 서명합니다. 다음 명령은 `crt` 형식으로도 인증서를 출력합니다.

```
openssl x509 -req -in server-saml.csr -days 3650 -CA
rootCACert-saml.pem -CAkey rootCAKey-saml.pem -
CAcreateserial -out server-saml.crt
```

e. 키 파일을 RSA로 변환합니다. Tableau에서 SAML을 사용하려면 RSA 키 파일이 필요합니다. 키를 변환하려면 다음 명령을 실행합니다.

```
openssl rsa -in server-saml.key -out server-saml-rsa.key
```

3. SAML을 구성합니다. 엔터티 ID와 반환 URL, 메타데이터 파일, 인증서 파일 및 키 파일에 대한 경로를 지정하여 다음 명령을 실행합니다.

```
tsm authentication saml configure --idp-entity-id
"https://tableau.example.com" --idp-return-url
"https://tableau.example.com" --idp-metadata idp_metadata.xml -
-cert-file "server-saml.crt" --key-file "server-saml-rsa.key"

tsm authentication saml enable
```

4. 조직에서 Tableau Desktop 2021.4 이상을 실행하는 경우 다음 명령을 실행하여 역방향 프록시 서버를 통한 인증을 사용하도록 설정해야 합니다.

   최상위 수준 도메인 쿠키 유지를 허용하도록 사전 인증 모듈(예: Mellon)을 구성한 경우 Tableau Desktop 버전 2021.2.1 ~ 2021.3은 이 명령을 실행하지 않고 작동합니다.

```
tsm configuration set -k features.ExternalBrowserOAuth -v false
```

5. 구성 변경 내용을 적용합니다.

```
tsm pending-changes apply
```

tsig-httpd 서비스 다시 시작

Tableau Server 배포에서 변경 내용을 적용하면 Tableau Server 독립 게이트웨이 컴퓨터에 다시 로그인하고 다음 명령을 실행하여 tsig-httpd 서비스를 다시 시작합니다.

```
sudo su - tableau-tsig
systemctl --user restart tsig-httpd
exit
```

SAML 기능 확인

전체 SAML 기능을 확인하려면 이 절차를 시작할 때 만든 Tableau 관리자 계정으로 공용 URL(예: https://tableau.example.com)을 사용하여 Tableau Server에 로그인합니다.

TSM이 시작되지 않거나("게이트웨이 오류") 연결을 시도할 때 브라우저 오류가 발생하는 경우 Tableau Server 독립 게이트웨이 문제 해결을 참조하십시오.

두 번째 인스턴스의 독립 게이트웨이에서 인증 모듈 구성

독립 게이트웨이의 첫 번째 인스턴스를 성공적으로 구성한 후 두 번째 인스턴스를 배포합니다. 여기에 나온 예제는 이 항목에 설명된 AWS/Mellon/Okta 시나리오를 설치하는 마지막 프로세스입니다. 이 절차는 이전에 이 항목(독립 게이트웨이 설치)에 설명된 대로 두 번째 인스턴스에 독립 게이트웨이를 이미 설치한 것으로 가정합니다.

두 번째 독립 게이트웨이 배포 프로세스에는 다음 단계가 필요합니다.

1. 독립 게이트웨이의 두 번째 인스턴스에서: Mellon 인증 모듈을 설치합니다.

   이 항목의 앞부분에서 설명한 대로 Mellon 인증 모듈을 구성하지 마십시오. 대신 다음 단계에 설명된 대로 구성을 복제해야 합니다.

2. 구성된(첫 번째) 독립 게이트웨이 인스턴스에서:

   기존 Mellon 구성에 대한 tar 복사본을 만듭니다. tar 백업에는 모든 디렉터리 계층 및 사용 권한이 보존됩니다. 다음 명령을 실행합니다.

   ```
   cd /etc
   sudo tar -cvf mellon.tar mellon
   ```

mellon.tar를 독립 게이트웨이의 두 번째 인스턴스에 복사합니다.

3. 독립 게이트웨이의 두 번째 인스턴스에서:

   /etc 디렉터리의 두 번째 인스턴스에 tar 파일을 추출("압축 풀기")합니다. 다음 명령을 실행합니다.

   ```
   cd /etc

   sudo tar -xvf mellon.tar
   ```

4. Tableau Server 배포의 노드 1에서: 두 번째 독립 게이트웨이의 연결 정보로 연결 파일(tsig.json)을 업데이트합니다. 이전에 이 항목(독립 게이트웨이 설치)에 설명된 대로 인증 키를 검색해야 합니다.

   아래에 예제 연결 파일(tsig.json)이 나와 있습니다.

   ```
   {
   "independentGateways": [
    {
      "id": "ip-10-0-1-169.ec2.internal",
      "host": "ip-10-0-1-169.ec2.internal",
      "port": "21319",
      "protocol" : "http",
      "authsecret": "13660-27118-29070-25482-9518-22453"
    },
    {
      "id": "ip-10-0-2-230.ec2.internal",
      "host": "ip-10-0-2-230.ec2.internal",
      "port": "21319",
      "protocol" : "http",
      "authsecret": "9055-27834-16487-27455-30409-7292"
    }]
    }
   ```

5. Tableau Server 배포의 노드 1에서: 다음 명령을 실행하여 구성을 업데이트합니다.

```
tsm stop

tsm topology external-services gateway update -c tsig.json

tsm start
```

6. Tableau Server가 시작될 때 독립 게이트웨이의 두 인스턴스에서: `tsig-httpd` 프로세스를 다시 시작합니다.

```
sudo su - tableau-tsig

systemctl --user restart tsig-httpd

exit
```

7. AWS **EC2>대상 그룹**에서: 두 번째 독립 게이트웨이 인스턴스를 실행하는 EC2 인스턴스를 포함하도록 대상 그룹을 업데이트합니다.

   방금 만든 대상 그룹을 선택한 다음 대상 탭을 클릭합니다.

   - **편집**을 클릭합니다.
   - 두 번째 독립 게이트웨이 컴퓨터의 EC2 인스턴스를 선택한 다음 **등록된 항목에 추가**를 클릭하고 **저장**을 클릭합니다.

6부 - 설치 후 구성

부하 분산 장치에서 Tableau Server로의 SSL/TLS 구성

일부 조직의 경우 클라이언트에서 백엔드 서비스에 이르는 엔드 투 엔드 암호화 채널을 요구합니다. 지금까지 설명한 기본 참조 아키텍처는 클라이언트에서 부하 분산 장치(조직의 웹 계층에서 실행됨)로의 SSL을 구성할 것을 명시합니다.

이 섹션에서는 AWS 참조 아키텍처 예제에서 Tableau Server와 독립 게이트웨이에 대한 SSL/TLS를 구성하는 방법을 설명합니다. AWS 참조 아키텍처에서 Apache에 SSL/TLS를 구성하는 방법을 설명하는 구성 예제는 예제: AWS 참조 아키텍처에서 SSL/TLS 구성을 참조하십시오.

지금은 8000~9000 범위에서 실행되는 백엔드 Tableau Server 프로세스에 대해 TLS가 지원되지 않습니다. TLS를 사용하려면 Tableau Server에 대한 릴레이 연결로 독립 게이트웨이를 구성해야 합니다.

이 절차는 독립 게이트웨이와 Tableau Server 간의 연결에 TLS를 사용하고 구성하는 방법을 설명합니다. 이 절차에서는 HTTPS/443을 통해 릴레이 트래픽을 암호화하고 HTTPS/21319를 통해 하우스키핑 트래픽을 암호화합니다.

이 예 전반에 걸친 Linux 절차는 RHEL와 같은 배포에 대한 명령을 보여줍니다. 특히 여기에 나온 명령은 Amazon Linux 2 배포를 통해 개발되었습니다. Ubuntu 배포를 실행 중인 경우 그에 따라 명령을 편집합니다.

여기의 지침은 이 가이드에 나온 특정 AWS 예제 참조 아키텍처에 권장되는 것입니다. 따라서 선택적 구성은 포함되어 있지 않습니다. 전체 참조 설명서는 *독립 게이트웨이에서 TLS 구성*(Linux)을 참조하십시오.

TLS를 구성하기 전에

업무 시간이 아닐 때 TLS 구성을 수행합니다. 구성할 때는 Tableau Server를 한 번 이상 다시 시작해야 합니다. 전체 4노드 참조 아키텍처 배포를 실행하는 경우 다시 시작하는 데 시간이 걸릴 수 있습니다.

- 클라이언트에서 HTTP를 통해 Tableau Server에 연결할 수 있는지 확인합니다. 독립 게이트웨이를 통해 TLS를 구성하는 프로세스는 여러 단계를 포함하며 문제 해결이 필요할 수 있습니다. 따라서 TLS를 구성하기 전에 완벽하게 작동하는 Tableau Server 배포로 시작하는 것이 좋습니다.

- TLS/SSL 인증서, 키 및 관련 자산을 수집합니다. 독립 게이트웨이 및 Tableau Server에 대한 SSL 인증서가 필요합니다. 인증서 관리 및 배포를 간소화하고 보안을 위한 모범 사례를 적용하기 위해 신뢰할 수 있는 주요 외부 CA(인증 기관)에서 생성된 인증서를 사용할 것을 권장합니다. 또는 자체 서명 인증서를 생성하거나 TLS용 PKI의 인증서를 사용할 수도 있습니다.

  이 항목의 구성 예에서는 설명을 위해 다음 자산 이름을 사용합니다.
  - `tsig-ssl.crt`: 독립 게이트웨이의 TLS/SSL 인증서입니다.
  - `tsig-ssl.key`: 독립 게이트웨이의 `tsig-ssl.crt` 관련 개인 키입니다.
  - `ts-ssl.crt`: Tableau Server용 TLS/SSL 인증서입니다.
  - `ts-ssl.key`: Tableau Server의 `tsig-ssl.crt` 관련 개인 키입니다.
  - `tableau-server-CA.pem`: Tableau Server 컴퓨터에 대한 인증서를 생성하는 CA의 루트 인증서입니다. 신뢰할 수 있는 주요 타사의 인증서를 사용하는 경우에는 일반적으로 이 인증서가 필요하지 않습니다.
  - `rootTSIG-CACert.pem`: 독립 게이트웨이 컴퓨터에 대한 인증서를 생성하는 CA의 루트 인증서입니다. 신뢰할 수 있는 주요 타사의 인증서를 사용하는 경우에는 일반적으로 이 인증서가 필요하지 않습니다.
  - SAML에 필요한 다른 인증서 및 키 파일 자산이 있습니다. 해당 정보는 이 가이드의 5부에 나와 있습니다.
  - 구현 시 인증서 체인 파일을 사용해야 하는 경우 기술 자료 문서 인증서 체인이 있는 인증서를 사용할 때 독립 게이트웨이에서 TLS 구성(영문)을 참조하십시오.

- IdP에 액세스할 수 있는지 확인합니다. IdP를 인증에 사용하는 경우 SSL/TLS를 구성한 후 IdP에서 수신자 및 대상 URL을 변경해야 할 가능성이 높습니다.

TLS에 대해 독립 게이트웨이 컴퓨터 구성

TLS를 구성하는 프로세스에서는 오류가 많이 발생합니다. 독립 게이트웨이의 두 인스턴스에서 문제를 해결하려면 많은 시간이 소모될 수 있으므로 EDG 배포에서 독립 게이트웨이 1개로 TLS를 사용하도록 설정하고 구성하는 것이 좋습니다. 배포에서 TLS가 작동하는 것을 확인한 후 두 번째 독립 게이트웨이 컴퓨터를 구성합니다.

1단계: 독립 게이트웨이 컴퓨터에 인증서 및 키 배포

tsig-httpd 사용자에게 파일에 대한 읽기 액세스 권한이 있기만 하면 임의의 모든 디렉터리에 자산을 배포해도 됩니다. 이러한 파일의 경로는 다른 절차에 언급되어 있습니다. 이 항목에서는 아래에 표시된 것과 같이 `/etc/ssl`의 예제 경로를 사용합니다.

1. 개인 키에 대한 디렉터리를 만듭니다.

   ```
   sudo mkdir -p /etc/ssl/private
   ```

2. 인증서와 키 파일을 `/etc/ssl` 경로에 복사합니다. 예를 들면 다음과 같습니다.

   ```
   sudo cp tsig-ssl.crt /etc/ssl/certs/
   sudo cp tsig-ssl.key /etc/ssl/private/
   ```

3. (선택 사항) 자체 서명 또는 PKI 인증서를 Tableau Server의 SSL/TLS에 사용하는 경우 CA 루트 인증서 파일을 독립 게이트웨이 컴퓨터에도 복사해야 합니다. 예를 들면 다음과 같습니다.

   ```
   sudo cp tableau-server-CA.pem /etc/ssl/certs/
   ```

2단계: TLS의 환경 변수 업데이트

독립 게이트웨이 구성에 대한 포트 및 프로토콜 환경 변수를 업데이트해야 합니다.

다음과 같이 `/etc/opt/tableau/tableau_tsig/environment.bash` 파일을 업데이트하여 이러한 값을 변경합니다.

```
TSIG_HK_PROTOCOL="https"
TSIG_PORT="443"
TSIG_PROTOCOL="https"
```

3단계: HK 프로토콜에 대한 stub 구성 파일 업데이트

stub 구성 파일(`/var/opt/tableau/tableau_tsig/config/httpd.conf.stub`)을 수동으로 업데이트하여 HK(하우스키핑) 프로토콜에 대한 TLS 관련 Apache httpd 지시문을 설정합니다.

stub 구성 파일에는 `#TLS#` 마커로 주석 처리된 TLS 관련 지시문 블록이 포함되어 있습니다. 아래 예제에 표시된 것과 같이 지시문에서 마커를 제거하십시오. 이 예제에서는 `SSLCACertificateFile` 옵션을 사용하여 Tableau Server에 사용된 SSL 인증서에 루트 CA 인증서를 사용합니다.

```
#TLS# SSLPassPhraseDialog exec:/path/to/file
<VirtualHost *:${TSIG_HK_PORT}>
SSLEngine on
#TLS# SSLHonorCipherOrder on
#TLS# SSLCompression off
SSLCertificateFile /etc/ssl/certs/tsig-ssl.crt
SSLCertificateKeyFile /etc/ssl/private/tsig-ssl.key
SSLCACertificateFile /etc/ssl/certs/tableau-server-CA.pem
#TLS# SSLCARevocationFile /path/to/file
</VirtualHost>
```

독립 게이트웨이를 다시 설치하면 이러한 변경 내용이 손실됩니다. 따라서 백업 복사본을 만드는 것이 좋습니다.

4단계: stub 파일 복사 및 서비스 다시 시작

1. 마지막 단계에서 업로드한 파일을 복사하여 httpd.conf를 변경 내용으로 업데이트합니다.

   ```
   cp /var/opt/tableau/tableau_tsig/config/httpd.conf.stub
   /var/opt/tableau/tableau_tsig/config/httpd.conf
   ```

2. 독립 게이트웨이 서비스를 다시 시작합니다.

```
sudo su - tableau-tsig
systemctl --user restart tsig-httpd
exit
```

다시 시작한 후 Tableau Server에서 다음 집합의 단계를 실행하기 전까지는 독립 게이트웨이가 작동하지 않습니다. Tableau Server에서 단계를 완료하면 독립 게이트웨이가 변경 내용을 가져오고 온라인으로 전환됩니다.

TLS에 대해 Tableau Server 노드 1 구성

Tableau Server 배포의 노드 1에서 다음 단계를 실행합니다.

1단계: 인증서 및 키를 복사하고 TSM을 중지

1. Tableau Server "외부 SSL" 인증서와 키가 노드 1에 복사되었는지 확인합니다.

2. 가동 중단을 최소화하려면 TSM을 중지하고 다음 단계를 실행한 다음 변경 내용이 적용된 후 TSM을 시작하는 것이 좋습니다.

```
tsm stop
```

2단계: 인증서 자산을 설정하고 독립 게이트웨이 구성을 사용하도록 설정

1. 독립 게이트웨이에 대한 인증서 및 키 파일의 위치를 지정합니다. 이러한 경로는 독립 게이트웨이 컴퓨터의 위치를 참조합니다. 참고로 이 예제에서는 HTTPS 및 하우스키핑 트래픽 보호에 동일한 인증서 및 키 쌍이 사용된다고 가정합니다.

```
tsm configuration set -k gateway.tsig.ssl.cert.file_name -v
/etc/ssl/certs/tsig-ssl.crt --force-keys
tsm configuration set -k gateway.tsig.ssl.key.file_name -v
/etc/ssl/private/tsig-ssl.key --force-keys
```

2. 독립 게이트웨이의 HTTPS 및 HK 프로토콜에 TLS를 사용하도록 설정합니다.

```
tsm configuration set -k gateway.tsig.ssl.enabled -v true --
force-keys
tsm configuration set -k gateway.tsig.hk.ssl.enabled -v true --
force-keys
```

3. (선택 사항) 자체 서명 또는 PKI 인증서를 독립 게이트웨이의 SSL/TLS에 사용하는 경우 CA 루트 인증서 파일을 업로드해야 합니다. CA 루트 인증서 파일은 독립 게이트웨이 컴퓨터에 대한 인증서를 생성하는 데 사용된 루트 인증서입니다. 예를 들면 다음과 같습니다.

   ```
   tsm security custom-cert add -c rootTSIG-CACert.pem
   ```

4. (선택 사항) 자체 서명 또는 PKI 인증서를 Tableau Server의 SSL/TLS에 사용하는 경우 CA 루트 인증서 파일을 독립 게이트웨이의 `/etc/ssl/certs` 디렉터리에도 복사해야 합니다. CA 루트 인증서 파일은 Tableau Server 컴퓨터에 대한 인증서를 생성하는 데 사용된 루트 인증서입니다. 독립 게이트웨이에 인증서를 복사한 후 다음 tsm 명령을 사용하여 노드 1의 인증서 위치를 지정해야 합니다. 예를 들면 다음과 같습니다.

   ```
   tsm configuration set -k gateway.tsig.ssl.proxy.gateway_relay_
   cluster.cacertificatefile -v /etc/ssl/certs/tableau-server-
   CA.pem --force-keys
   ```

5. (선택 사항: 테스트 전용) 자체 서명 또는 PKI 인증서를 여러 컴퓨터에서 공유하기 때문에 인증서의 주체 이름이 컴퓨터 이름과 일치하지 않는 경우 인증서 유효성 검사를 사용하지 않도록 설정해야 합니다.

   ```
   tsm configuration set -k gateway.tsig.ssl.proxy.verify -v
   optional_no_ca --force-keys
   ```

3단계: Tableau Server에 "외부 SSL"을 사용하도록 설정하고 변경 내용을 적용

1. Tableau Server에서 "외부 SSL"을 사용하도록 설정하고 구성합니다.

   ```
   tsm security external-ssl enable --cert-file ts-ssl.crt --key-
   file ts-ssl.key
   ```

2. 변경 내용을 적용합니다.

```
tsm pending-changes apply
```

4단계: 게이트웨이 구성 JSON 파일을 업데이트하고 tsm을 시작

1. Tableau Server 측의 독립 게이트웨이 구성 파일(예: `tsig.json`)을 업데이트하여 독립 게이트웨이 개체에 대한 `https` 프로토콜을 지정합니다.

```
"protocol" : "https",
```

2. 독립 게이트웨이의 두 번째 인스턴스에 대한 연결 정보를 제거(또는 주석 처리)합니다. JSON을 저장하기 전에 외부 편집기에서 이를 확인하십시오.

   독립 게이트웨이 인스턴스 1개에 대한 TLS를 구성하고 확인한 후 독립 게이트웨이의 두 번째 인스턴스에 대한 연결 정보로 이 JSON 파일을 업데이트합니다.

3. 다음 명령을 실행하여 독립 게이트웨이 구성을 업데이트합니다.

```
tsm topology external-services gateway update -c tsig.json
```

4. TSM을 시작합니다.

```
tsm start
```

5. TSM을 시작하는 동안 독립 게이트웨이 인스턴스에 로그인하고 `tsig-httpd` 서비스를 다시 시작합니다.

```
sudo su - tableau-tsig
systemctl --user restart tsig-httpd
exit
```

IdP 인증 모듈 URL을 HTTPS로 업데이트

Tableau에 대한 외부 ID 공급자를 구성한 경우 IdP 관리 대시보드의 반환 URL을 업데이트해야 할 수 있습니다.

예를 들어 Okta 사전 인증 응용 프로그램을 사용하는 경우 수신자 URL 및 대상 URL에 HTTPS 프로토콜을 사용하도록 응용 프로그램을 업데이트해야 합니다.

HTTPS에 대한 AWS 부하 분산 장치 구성

이 가이드에 설명된 대로 AWS 부하 분산 장치를 배포하는 경우 독립 게이트웨이를 실행하는 컴퓨터로 HTTPS 트래픽을 보내도록 AWS 부하 분산 장치를 다시 구성해야 합니다.

1. 기존 HTTP 대상 그룹을 삭제합니다.

   대상 그룹에서 부하 분산 장치에 구성된 HTTP 대상 그룹을 선택하고 **동작**을 클릭한 다음 **삭제**를 클릭합니다.

2. HTTPS 대상 그룹을 만듭니다.

   대상 그룹 > 대상 그룹 만들기

   - "인스턴스" 선택
   - 대상 그룹 이름 입력(예: `TG-internal-HTTPS`)
   - VPC 선택
   - 프로토콜: HTTPS 443
   - **상태 확인 > 고급 상태 확인 설정 > 성공 코드**에서 읽을 코드 목록(`200,303`)을 추가합니다.
   - **만들기**를 클릭합니다.

3. 방금 만든 대상 그룹을 선택한 다음 **대상** 탭을 클릭합니다.

   - **편집**을 클릭합니다.
   - 이전에 구성하여 Tableau Server 독립 게이트웨이를 실행 중인 EC2 인스턴스를 선택한 다음 **등록된 항목에 추가**를 클릭합니다.
   - **저장**을 클릭합니다.

4. 대상 그룹이 만들어지면 연결 유지를 사용하도록 설정해야 합니다.

   - AWS 대상 그룹 페이지(**EC2 > 부하 분산 > 대상 그룹**)를 열고 방금 설정한 대상 그룹 인스턴스를 선택합니다. **동작** 메뉴에서 **특성 편집**을 선택합니다.

- **특성 편집** 페이지에서 **연결 유지**를 선택하고 기간을 `1 day`로 지정한 다음 **변경 내용을 저장**합니다.

5. 부하 분산 장치에서 수신기 규칙을 업데이트합니다. 이 배포에 대해 구성한 부하 분산 장치를 선택한 다음 **수신기** 탭을 클릭합니다.

   - **HTTP:80**에서 **규칙 보기/편집**을 클릭합니다. 결과로 표시되는 **규칙** 페이지에서 편집 아이콘을 클릭하여 규칙을 편집합니다(페이지 맨 위에서 한 번 클릭하여 편집한 다음 규칙별로 다시 편집). 기존 THEN 규칙을 삭제하고 **동작 추가 > 리디렉션 대상...**을 클릭하여 바꿉니다. 결과로 나온 THEN 구성에서 `HTTPS` 및 포트 `443`을 지정하고 다른 옵션을 기본 설정으로 유지합니다. 설정을 저장한 다음 **업데이트**를 클릭합니다.
   - **HTTPS:443**에서 **규칙 보기/편집**을 클릭합니다. 결과로 표시되는 **규칙** 페이지에서 편집 아이콘을 클릭하여 규칙을 편집합니다(페이지 맨 위에서 한 번 클릭하여 편집한 다음 규칙별로 다시 편집). 기존 THEN 규칙을 삭제하고 **동작 추가 > 전달 대상...**을 클릭하여 바꿉니다. 대상 그룹을 방금 만든 HTTPS 그룹으로 지정합니다. **그룹 수준 연결 유지**에서 연결 유지를 사용하도록 설정하고 기간을 1일로 설정합니다. 설정을 저장한 다음 **업데이트**를 클릭합니다.

6. 부하 분산 장치에서 유휴 시간 초과를 400초로 업데이트합니다. 이 배포에 대해 구성한 부하 분산 장치를 선택한 다음 **동작 > 특성 편집**을 클릭합니다. **유휴 시간 초과**를 `400` 초로 설정한 다음 **저장**을 클릭합니다.

TLS 확인

TLS 기능을 확인하려면 이 절차를 시작할 때 만든 Tableau 관리자 계정으로 공용 URL (예: https://tableau.example.com)을 사용하여 Tableau Server에 로그인합니다.

TSM이 시작되지 않거나 다른 오류가 발생하는 경우 Tableau Server 독립 게이트웨이 문제 해결을 참조하십시오.

SSL에 대해 독립 게이트웨이의 두 번째 인스턴스 구성

독립 게이트웨이의 첫 번째 인스턴스를 성공적으로 구성한 후 두 번째 인스턴스를 배포합니다.

두 번째 독립 게이트웨이 배포 프로세스에는 다음 단계가 필요합니다.

1. 구성된(첫 번째) 독립 게이트웨이 인스턴스에서: 두 번째 독립 게이트웨이 인스턴스의 해당 위치에 다음 파일을 복사합니다.
   - `/etc/ssl/certs/tsig-ssl.crt`
   - `/etc/ssl/private/tsig-ssl.key`(두 번째 인스턴스에서 `private` 디렉터리를 만들어야 함)
   - `/var/opt/tableau/tableau_tsig/config/httpd.conf.stub`
   - `/etc/opt/tableau/tableau_tsig/environment.bash`
2. Tableau Server 배포의 노드 1에서: 두 번째 독립 게이트웨이의 연결 정보로 연결 파일(`tsig.json`)을 업데이트합니다.

   아래에 예제 연결 파일(`tsig.json`)이 나와 있습니다.

```
{
"independentGateways": [
 {
   "id": "ip-10-0-1-169.ec2.internal",
   "host": "ip-10-0-1-169.ec2.internal",
   "port": "21319",
   "protocol" : "https",
   "authsecret": "13660-27118-29070-25482-9518-22453"
 },
 {
   "id": "ip-10-0-2-230.ec2.internal",
   "host": "ip-10-0-2-230.ec2.internal",
   "port": "21319",
```

```
    "protocol" : "https",
    "authsecret": "9055-27834-16487-27455-30409-7292"
  }]
  }
```

3. Tableau Server 배포의 노드 1에서: 다음 명령을 실행하여 구성을 업데이트합니다.

   ```
   tsm stop
   ```

   ```
   tsm topology external-services gateway update -c tsig.json
   ```

   ```
   tsm start
   ```

4. 독립 게이트웨이의 두 인스턴스에서: Tableau Server가 시작될 때 독립 게이트웨이의 두 인스턴스에서 `tsig-httpd` 프로세스를 다시 시작합니다.

   ```
   sudo su - tableau-tsig
   ```

   ```
   systemctl --user restart tsig-httpd
   ```

   ```
   exit
   ```

5. AWS **EC2>대상 그룹**에서: 두 번째 독립 게이트웨이 인스턴스를 실행하는 EC2 인스턴스를 포함하도록 대상 그룹을 업데이트합니다.

   방금 만든 대상 그룹을 선택한 다음 대상 탭을 클릭합니다.

   - **편집**을 클릭합니다.
   - 두 번째 독립 게이트웨이 컴퓨터의 EC2 인스턴스를 선택한 다음 **등록된 항목에 추가**를 클릭하고 **저장**을 클릭합니다.

Postgres에 대한 SSL 구성

필요한 경우 Tableau Server의 외부 리포지토리 연결을 위한 Postgres 연결에 사용할 SSL(TLS)을 구성할 수 있습니다.

인증서 관리 및 배포를 간소화하고 보안을 위한 모범 사례를 적용하기 위해 신뢰할 수 있는 주요 외부 CA(인증 기관)에서 생성된 인증서를 사용할 것을 권장합니다. 또는 자체 서명 인증서를 생성하거나 TLS용 PKI의 인증서를 사용할 수도 있습니다.

이 절차에서는 예제 AWS 참조 아키텍처에서 RHEL 유사 Linux 배포의 Postgres 호스트에서 OpenSSL을 사용하여 자체 서명 인증서를 생성하는 방법에 대해 설명합니다.

SSL 인증서를 생성하고 서명한 후에 CA 인증서를 Tableau 호스트에 복사해야 합니다.

Postgress를 실행하는 호스트에서 다음을 수행합니다.

1. 서명 루트 CA(인증 기관)키를 생성합니다.

   ```
   openssl genrsa -out pgsql-rootCAKey.pem 2048
   ```

2. 루트 CA 인증서를 만듭니다.

   ```
   openssl req -x509 -sha256 -new -nodes -key pgsql-rootCAKey.pem
   -days 3650 -out pgsql-rootCACert.pem
   ```

   인증서 필드에 값을 입력하라는 메시지가 표시됩니다. 예:

   ```
   Country Name (2 letter code) [XX]:US
   State or Province Name (full name) []:Washington
   Locality Name (eg, city) [Default City]:Seattle
   Organization Name (eg, company) [Default Company Ltd]:Tableau
   Organizational Unit Name (eg, section) []:Operations
   Common Name (eg, Postgres server's hostname) []:ip-10-0-1-
   189.us-west-1.compute.internal
   Email Address []:example@tableau.com
   ```

3. Postgres 컴퓨터에 대한 인증서 및 관련 키(아래 예에서 `server.csr` 및 `server.key`)를 만듭니다. 인증서의 주체 이름은 Postgres 호스트의 EC2 사설 DNS 이름과 일치해야 합니다. 주체 이름은 `-subj` 옵션과 `"/CN=<private DNS name>"` 형식을 사용하여 설정됩니다. 예:

   ```
   openssl req -new -nodes -text -out server.csr -keyout
   server.key -subj "/CN=ip-10-0-1-189.us-west-1.compute.internal"
   ```

4. 2단계에서 만든 CA 인증서를 사용하여 새 인증서에 서명합니다. 다음 명령은 `crt` 형식으로도 인증서를 출력합니다.

```
openssl x509 -req -in server.csr -days 3650 -CA pgsql-
rootCACert.pem -CAkey pgsql-rootCAKey.pem -CAcreateserial -out
server.crt
```

5. crt 및 key 파일을 Postgres `/var/lib/pgsql/13/data/` 경로에 복사합니다.

```
sudo cp server.crt /var/lib/pgsql/13/data/
```

```
sudo cp server.key /var/lib/pgsql/13/data/
```

6. 루트 사용자로 전환합니다.

```
sudo su
```

7. 인증서 및 키 파일에 대한 사용 권한을 설정합니다. 다음 명령을 실행합니다.

```
cd /var/lib/pgsql/13/data
chown postgres.postgres server.crt
chown postgres.postgres server.key
chmod 0600 server.crt
chmod 0600 server.key
```

8. pg_haba 구성 파일(`/var/lib/pgsql/13/data/pg_hba.conf`)을 업데이트하여 md5 트러스트를 지정합니다.

   기존 연결 문을 변경합니다. 변경 전:

   `host all all 10.0.30.0/24 password` 및

   `host all all 10.0.31.0/24 password`

   변경 후:

   `host all all 10.0.30.0/24 md5` 및

   `host all all 10.0.31.0/24 md5.`

9. 다음 줄을 추가하여 postgresql 파일 (`/var/lib/pgsql/13/data/postgresql.conf`)을 업데이트합니다.

```
ssl = on
```

10. 루트 사용자 모드를 끝냅니다.

```
exit
```

11. Postgres를 다시 시작합니다.

```
sudo systemctl restart postgresql-13
```

선택 사항: Tableau Server에서 Postgres SSL용 인증서 신뢰 유효성 검사 사용

4부 - Tableau Server 설치 및 구성의 설치 절차를 수행한 경우 Postgres 연결에 대해 선택적인 SSL을 사용하여 Tableau Server가 구성됩니다. 즉, 앞서 설명한 것처럼 Postgres에 SSL을 구성하면 연결이 암호화됩니다.

연결에 대해 인증서 신뢰 유효성 검사를 요구하려면 Tableau Server에서 다음 명령을 실행하여 Postgres 호스트 연결을 재구성해야 합니다.

```
tsm topology external-services repository replace-host -f
<filename>.json -c CACert.pem
```

여기서 `<filename>.json`은 외부 Postgres 구성에 명시된 연결 파일입니다. 그리고 `CACert.pem`은 Postgres에서 사용하는 SSL/TLS 인증서의 CA 인증서 파일입니다.

선택 사항: SSL 연결 확인

SSL 연결을 확인하려면 다음을 수행해야 합니다.

- Tableau Server 노드 1에 Postgres 클라이언트를 설치합니다.
- 이전 절차에서 만든 루트 인증서를 Tableau 호스트에 복사합니다.
- 노드 1에서 Postgres 서버에 연결합니다.

노드 1에 Postgres 클라이언트 설치

이 예에서는 Postgres 13.4 버전을 설치하는 방법을 보여 줍니다. 외부 리포지토리에 대해 실행 중인 것과 동일한 버전을 설치해야 합니다.

1. 노드 1에서 `/etc/yum.repos.d` 경로에 pgdg.repo 파일을 만들고 편집합니다. 다음 구성 정보로 파일을 채웁니다.

   ```
   [pgdg13]
   name=PostgreSQL 13 for RHEL/CentOS 7 - x86_64

   baseurl=https://download.postgresql.org/pub/repos/yum/13/redhat-
   /rhel-7-x86_64
   enabled=1
   gpgcheck=0
   ```

2. Postgres 클라이언트를 설치합니다.

   ```
   sudo yum install postgresql13-13.4-1PGDG.rhel7.x86_64
   ```

노드 1에 루트 인증서 복사

CA 인증서(`pgsql-rootCACert.pem`)를 Tableau 호스트에 복사합니다.

```
scp ec2-user@<private-DNS-name-of-Postgress-host>:/home/ec2-
user/pgsql-rootCACert.pem /home/ec2-user
```

노드 1에서 SSL을 통해 Postgres 호스트에 연결

Postgres 서버 호스트 IP 주소와 루트 CA 인증서를 지정하여 노드 1에서 다음 명령을 실행합니다.

```
psql "postgresql://postgres@<IP-
address>:5432/postgres?sslmode=verify-ca&sslrootcert=pgsql-
rootCACert.pem"
```

예:

```
psql
"postgresql://postgres@10.0.1.189:5432/postgres?sslmode=verify-
ca&sslrootcert=pgsql-rootCACert.pem"
```

Postgres에 비밀번호를 입력하라는 메시지가 표시됩니다. 성공적으로 로그인되면 셸에서 다음을 반환합니다.

```
psql (13.4)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-
SHA384, bits: 256, compression: off)
Type "help" for help.
postgres=#
```

SMTP 및 이벤트 알림 구성

Tableau Server는 관리자와 사용자에게 이메일 알림을 보냅니다. 이메일 알림을 사용하도록 설정하려면 이메일 서버로 메일을 보내도록 Tableau Server를 구성해야 합니다. 또한 보내려는 이벤트 유형, 임계값 및 구독 정보도 지정해야 합니다.

SMTP 및 알림의 초기 구성에서는 아래의 구성 파일 템플릿을 사용하여 json 파일을 만드는 것이 좋습니다. 또한 *tsm configuration set*(Linux)에 설명되어 있는 구문을 사용하여 아래에 나와 있는 모든 단일 구성 키를 설정할 수 있습니다.

Tableau Server 배포의 노드 1에서 다음 절차를 실행합니다.

1. 다음 json 템플릿을 파일에 복사합니다. 조직의 SMTP 구성 옵션과 구독 및 알림으로 파일을 사용자 지정합니다.
   - 모든 SMTP 옵션의 목록 및 설명을 보려면 *SMTP CLI 구성 참조*(Linux)를 참조하십시오.
   - 모든 알림 이벤트 옵션의 목록 및 설명을 보려면 *서버 이벤트 알림 구성*(Linux)의 CLI 섹션을 참조하십시오.

```
{
"configKeys": {
       "svcmonitor.notification.smtp.server": "SMTP server host
name",
```

```
        "svcmonitor.notification.smtp.send_account": "SMTP user name",
        "svcmonitor.notification.smtp.port": 443,
        "svcmonitor.notification.smtp.password": "SMTP user account
password",
        "svcmonitor.notification.smtp.ssl_enabled": true,
        "svcmonitor.notification.smtp.from_address": "From email
address",
        "svcmonitor.notification.smtp.target_addresses": "To email
address1,address2",
        "svcmonitor.notification.smtp.canonical_url": "Tableau Server
URL",
        "backgrounder.notifications_enabled": true,
        "subscriptions.enabled": true,
        "subscriptions.attachments_enabled": true,
        "subscriptions.max_attachment_size_megabytes": 150,
        "svcmonitor.notification.smtp.enabled": true,
        "features.DesktopReporting": true,
        "storage.monitoring.email_enabled": true,
        "storage.monitoring.warning_percent": 20,
        "storage.monitoring.critical_percent": 15,
        "storage.monitoring.email_interval_min": 25,
        "storage.monitoring.record_history_enabled": true
        }
}
```

2. `tsm settings import -f file.json`을 실행하여 json 파일을 Tableau 서비스 관리자로 전달합니다.

3. `tsm pending-changes apply` 명령을 실행하여 변경 내용을 적용합니다.

4. `tsm email test-smtp-connection`을 실행하여 연결 구성을 보고 확인합니다.

PostgreSQL 드라이버 설치

Tableau Server에서 관리 뷰를 보려면 PostgreSQL 드라이버를 Tableau Server 배포의 노드 1에 설치해야 합니다.

1. Tableau 드라이버 다운로드 페이지로 이동하고 PostgreSQL jar 파일에 대한 URL을 복사합니다.

2. Tableau 배포의 각 노드에서 다음 절차를 실행합니다.

   - 다음 파일 경로를 만듭니다.

     ```
     sudo mkdir -p /opt/tableau/tableau_driver/jdbc
     ```

   - 새 경로에서 PostgreSQL jar 파일의 최신 버전을 다운로드합니다. 예:

     ```
     sudo wget
     https://downloads.tableau.com/drivers/linux/postgresql/pos-
     tgresql-42.2.22.jar
     ```

3. 초기 노드에서 Tableau Server를 다시 시작합니다.

   ```
   tsm restart
   ```

강력한 비밀번호 정책 구성

IdP 인증 솔루션을 사용하여 Tableau Server를 배포하지 않는 경우 기본 Tableau 비밀번호 정책의 보안을 강화하는 것이 좋습니다.

IdP를 사용하여 Tableau Server를 배포하는 경우 IdP를 통해 비밀번호 정책을 관리해야 합니다.

다음 절차에는 Tableau Server에서 비밀번호 정책을 설정하기 위한 json 구성이 포함되어 있습니다. 아래의 옵션에 대한 자세한 내용은 *로컬 인증*(Linux)을 참조하십시오.

1. 다음 json 템플릿을 파일에 복사합니다. 암호 정책 구성에 따라 키 값을 채웁니다.

   ```
   {
    "configKeys": {
          "wgserver.localauth.policies.mustcontainletters.enabled":
   true,
          "wgserver.localauth.policies.mustcontainuppercase.enabled":
   ```

```
true,
       "wgserver.localauth.policies.mustcontainnumbers.enabled":
true,
       "wgserver.localauth.policies.mustcontainsymbols.enabled":
true,
       "wgserver.localauth.policies.minimumpasswordlength.enabled":
true,
       "wgserver.localauth.policies.minimumpasswordlength.value": 12,
       "wgserver.localauth.policies.maximumpasswordlength.enabled":
false,
       "wgserver.localauth.policies.maximumpasswordlength.value":
255,
       "wgserver.localauth.passwordexpiration.enabled": true,
       "wgserver.localauth.passwordexpiration.days": 90,
       "wgserver.localauth.ratelimiting.maxbackoff.minutes": 60,
       "wgserver.localauth.ratelimiting.maxattempts.enabled": false,
       "wgserver.localauth.ratelimiting.maxattempts.value": 5,
       "features.PasswordReset": true
 }
}
```

2. `tsm settings import -f file.json`을 실행하여 json 파일을 Tableau 서비스 관리자로 전달하고 Tableau Server를 구성합니다.

3. `tsm pending-changes apply` 명령을 실행하여 변경 내용을 적용합니다.

7부 - 유효성 검사, 도구 및 문제 해결

이 부에는 설치 후 검증 단계 및 문제 해결 지침이 수록되어 있습니다.

장애 조치 시스템 검증

배포를 구성한 후에는 간단한 장애 조치 테스트를 실행하여 시스템 이중화를 검증하는 것이 좋습니다.

다음 단계를 실행하여 장애 조치 기능을 검증하는 것이 좋습니다.

1. 독립 게이트웨이(TSIG1)의 첫 번째 인스턴스를 종료합니다. 모든 인바운드 트래픽은 독립 게이트웨이(TSIG2)의 두 번째 인스턴스를 통해 라우팅되어야 합니다.
2. TSIG1을 다시 시작한 다음 TSIG2를 종료합니다. 모든 인바운드 트래픽은 TSIG1을 통해 라우팅되어야 합니다.
3. TSIG2를 다시 시작합니다.
4. Tableau Server 노드 1을 종료합니다. 모든 Vizportal/응용 프로그램 서비스 트래픽이 노드 2로 장애 조치됩니다.

> **참고:** 2022년 9월부터 Tableau Server 2021.4 이상의 특정 버전에서는 노드 1 고가용성이 저하됩니다. 노드 1이 중단되면 클라이언트 연결이 실패합니다. 이 문제는 다음 유지 관리 버전에서 해결되었습니다.
>
> - 2021.4.15 이상
> - 2022.1.11 이상
> - 2023.1.3 이상
>
> ATR 활성화를 사용하는 Tableau Server 설치에 초기 노드 오류가 발생한 후

> 72시간의 유예 기간을 가지려면 이러한 버전 중 하나를 설치하거나 업그레이드하십시오. 자세한 내용은 Tableau 기술 자료에서 ATR을 사용하는 Tableau Server HA에 초기 노드 오류 후 유예 기간이 없음을 참조하십시오.

5. 노드 1을 다시 시작하고 노드 2를 종료합니다. 모든 Vizportal/응용 프로그램 서비스 트래픽이 노드 1로 장애 조치됩니다.
6. 노드 2를 다시 시작합니다.

이 맥락에서 "종료" 또는 "다시 시작"은 응용 프로그램을 미리 정상적으로 종료하지 않고 운영 체제 또는 가상 컴퓨터를 끄는 방식으로 수행합니다. 이 작업의 목표는 하드웨어 또는 가상 컴퓨터 장애를 시뮬레이션하는 것입니다.

각 장애 조치 테스트의 최소 검증 단계는 사용자를 인증하고 기본적인 보기 작업을 수행하는 것입니다.

장애를 시뮬레이션한 후 로그인을 시도할 때 "잘못된 요청" 브라우저 오류가 발생할 수 있습니다. 브라우저에서 캐시를 지우는 경우에도 이 오류가 나타날 수 있습니다. 이 문제는 주로 브라우저가 이전 IdP 세션의 데이터를 캐싱할 때 발생합니다. 로컬 브라우저 캐시를 지워도 이 오류가 계속되면 다른 브라우저로 연결하여 Tableau 시나리오의 유효성을 검사하십시오.

초기 노드 자동 복구

Tableau Server 버전 2021.2.4 이상에는 스크립트 디렉터리(`/app/tableau_server/packages/scripts.<version>`)에 자동화된 초기 노드 복구 스크립트인 `auto-node-recovery`가 포함되어 있습니다.

초기 노드에 문제가 있고 노드 2에 중복 프로세스가 있는 경우 Tableau Server가 계속 실행된다는 보장이 없습니다. Tableau Server는 초기 노드 장애 발생 후 최장 72시간 동안 계속 실행될 수 있으며, 그 이후에는 라이선스 서비스가 부족하여 다른 프로세스에 영향을 미칠 수 있습니다. 이러한 경우 사용자는 초기 노드에 장애가 발생한 후에도 계속 로그인하고 자신의 콘텐츠를 보고 사용할 수 있지만 관리 컨트롤러에 액세스할 수 없기 때문에 Tableau Server를 다시 구성할 수 없습니다.

중복 프로세스로 구성된 경우에도 초기 노드에 장애가 발생한 후 Tableau Server가 계속 작동하지 않을 수 있습니다.

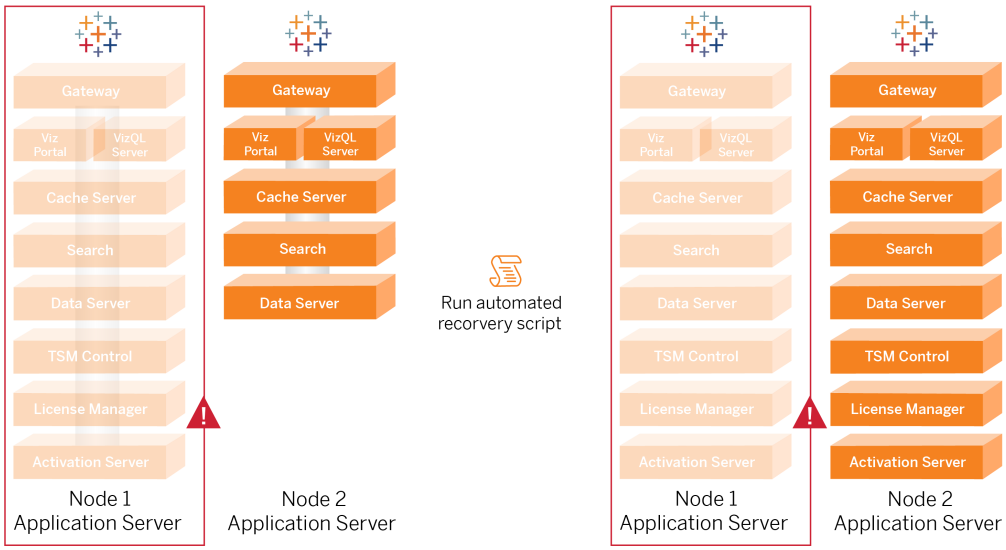


Tableau 초기 노드(노드 1) 오류를 복구하려면:

1. Tableau Server 노드 2에 로그인합니다.
2. 스크립트 디렉터리로 변경합니다.

   ```
   cd /app/tableau_server/packages/scripts.<version>
   ```

3. 다음 명령을 실행하여 스크립트를 시작합니다.

   ```
   sudo ./auto-node-recovery -p node1 -n node2 -k <license keys>
   ```

   여기서 `<license keys>`는 배포의 쉼표로 구분된(공백 없음) 라이선스 키 목록입니다. 라이선스 키에 액세스할 수 없는 경우 Tableau 고객 포털을 방문하여 해당 키를 검색하십시오. 예를 들면 다음과 같습니다.

   ```
   sudo ./auto-node-recovery -p node1 -n node2 -k TSB4-8675-309F-TW50-9RUS,TSNM-559N-ULL6-22VE-SIEN
   ```

auto-node-recovery 스크립트는 약 20단계를 실행하여 서비스를 노드 2로 복구합니다. 스크립트가 진행됨에 따라 각 단계가 터미널에 표시됩니다. 자세한 상태는 `/data/tableau_data/logs/app-controller-move.log`에 기록됩니다. 대부분의 환경에서 스크립트를 완료하는 데 35분에서 45분 정도 걸립니다.

초기 노드 복구 문제 해결

노드 복구가 실패하면 대화형으로 스크립트를 실행하여 유용한 프로세스에서 개별 단계를 허용하거나 허용하지 않을 수 있습니다. 예를 들어 스크립트가 프로세스 도중에 실패할 경우 로그 파일을 검토하고 구성을 변경한 다음 스크립트를 다시 실행할 수 있습니다. 대화형 모드로 실행하면 실패한 단계에 도달할 때까지 모든 단계를 건너뛸 수 있습니다.

대화형 모드로 실행하려면 script 인수에 `-i` 스위치를 추가합니다.

장애가 발생한 노드를 다시 구축

스크립트를 실행한 후에는 이전에 장애가 발생한 노드 1 호스트에 있었던 모든 서비스가 노드 2에서 실행됩니다. 4노드에서 추가하려면 부트스트랩 파일을 사용하여 새로운 Tableau Server 호스트를 배포하고 4부에 명시된 대로 원래 노드 2에 했던 것과 동일하게 구성합니다. 자세한 내용은 노드 2 구성을 참조하십시오.

switchto

switchto는 창 간 전환을 손쉽게 하는 Tim의 스크립트입니다.

1. 배스천 호스트의 홈 디렉터리에 있는 `switchto` 파일에 다음 코드를 복사합니다.

```
#!/bin/bash
#-----------------------------------------------------------------
-----
# switchto
#
# Helper function to simplify SSH into the various AWS hosts
```

```
when
# following the Tableau Server Enterprise Deployment Guide
(EDG).
#
# Place this file on your bastion host and provide your AWS
hosts'
# internal ip addresses or machine names here.
# Example: readonly NODE1="10.0.3.187"
#
readonly NODE1=""
readonly NODE2=""
readonly NODE3=""
readonly NODE4=""
readonly PGSQL=""
readonly PROXY1=""
readonly PROXY2=""

usage() {
echo "Usage: switchto.sh [ node1 | node2 | node3 | node4 |
pgsql | proxy1 | proxy2 ]"
}


ip=""

case $1 in
       node1)
               ip="$NODE1"
               ;;
       node2)
               ip="$NODE2"
               ;;
       node3)
               ip="$NODE3"
               ;;
```

```
        node4)
                ip="$NODE4"
                ;;
        pgsql)
                ip="$PGSQL"
                ;;
        proxy1)
                ip="$PROXY1"
                ;;
        proxy2)
                ip="$PROXY2"
                ;;
        ?)
                usage
                exit 0
                ;;
        *)
                echo "Unkown option $1."
                usage
                exit 1
                ;;
esac

if [[ -z $ip ]]; then
echo "You must first edit this file to provide the ip addresses
of your AWS hosts."
exit 1
fi

ssh -A ec2-user@$ip
```

2. 스크립트의 IP 주소를 업데이트하여 EC2 인스턴스에 매핑한 후 파일을 저장합니다.
3. 스크립트 파일에 사용 권한을 적용합니다.

```
sudo chmod +x switchto
```

사용법:

호스트를 전환하려면 다음 명령을 실행합니다.

```
./switchto <target>
```

예를 들어 노드 1로 전환하려면 다음 명령을 실행합니다.

```
./switchto node1
```

Tableau Server 독립 게이트웨이 문제 해결

Tableau Server에서 독립 게이트웨이, Okta, Mellon 및 SAML을 구성하는 프로세스에서는 오류가 많이 발생할 수 있습니다. 실패의 가장 일반적인 근본 원인은 문자열 오류입니다. 예를 들어 구성 중에 지정된 Okta URL의 후행 슬래시(/)로 인해 SAML 어설션 관련 불일치 오류가 발생할 수 있습니다. 이 오류는 한 가지 예입니다. 구성 중에 어떤 응용 프로그램에서든 잘못된 문자열을 입력하게 될 가능성이 많습니다.

tableau-tsig 서비스 다시 시작

항상 독립 게이트웨이 컴퓨터에서 tableau-tsig 서비스를 다시 시작하는 것으로 문제 해결 작업을 시작합니다. 이 서비스는 빠르게 다시 시작되며 다시 시작하면 Tableau Server에서 업데이트된 구성이 로드됩니다.

독립 게이트웨이 컴퓨터에서 다음 명령을 실행합니다.

```
sudo su - tableau-tsig
systemctl --user restart tsig-httpd
exit
```

잘못된 문자열 찾기

문자열 오류(복사/붙여넣기 실수, 문자열 잘림 등)가 있는 경우 구성한 각 설정을 검토하십시오.

- Okta 사전 인증 구성. 설정한 URL을 주의 깊게 검토합니다. 후행 슬래시가 있는지 확인합니다. HTTP와 HTTPS를 비교하여 확인합니다.
- 노드 1의 SAML 구성에 대한 셸 기록. 실행한 `tsm authentication saml configure` 명령을 검토합니다. 모든 URL이 Okta에 구성된 것과 일치하는지 확인합니다. 노드 1에서 셸 기록을 검토하는 동안 Mellon 구성 파일 경로를 지정하는 `tsm configuration set` 명령이 독립 게이트웨이에서 파일을 복사한 파일 경로에 정확히 매핑되는지 확인합니다.
- 독립 게이트웨이의 Mellon 구성. 셸 기록을 검토하여 Okta 및 Tableau SAML에서 구성한 것과 동일한 URL 문자열로 메타데이터를 만들었는지 확인합니다. `/etc/mellon/conf.d/global.conf`에 지정된 모든 경로가 올바르고 `MellonCookieDomain`이 Tableau 하위 도메인이 아닌 루트 도메인으로 설정되어 있는지 확인합니다.

관련 로그 검색

모든 문자열이 올바르게 설정된 것으로 보이면 로그에서 오류를 검사해야 합니다.

Tableau Server는 수십 개의 서로 다른 로그 파일에 오류 및 이벤트를 기록합니다. 독립 게이트웨이는 로컬 파일 집합에도 로그를 기록합니다. 이러한 로그를 검사할 때는 다음 순서를 따르는 것이 좋습니다.

독립 게이트웨이 로그 파일

독립 게이트웨이 로그 파일의 기본 위치는 `/var/opt/tableau/tableau_tsig/logs` 입니다.

- `access.log`: 이 로그는 Tableau Server 노드의 연결을 보여주는 항목이 있는 경우 유용합니다. TSM을 시작하려고 할 때 게이트웨이 오류가 발생해 시작되지 않는 경우 `access.log` 파일에 항목이 없다면 코어 연결 문제가 있는 것입니다. 항상 AWS 보안 그룹 구성을 첫 번째 단계로 확인합니다. 다른 일반적인 문제로는 `tsig.json`의 입력 오류가 있습니다. tsig.json을 업데이트하는 경우 `tsm topology external-services gateway update -c tsig.json`을 실행하기 전에 `tsm stop`을 실행합니다. tsig.json이 업데이트되면 `tsm start`를 실행합니다.
- `error.log`: 다른 항목 중에서 이 로그에는 SAML 및 Mellon 오류가 포함됩니다.

Tableau Server tabadminagent 로그 파일

`tabadminagent`(`tabadmincontroller`가 아님) 파일 집합은 독립 게이트웨이 관련 오류를 해결하는 것과 관련된 로그 파일만 모아 놓은 것입니다.

독립 게이트웨이 오류가 `tabdminagent`에 기록된 위치를 찾아야 합니다. 이러한 오류는 아무 노드에서나 발생할 수 있지만 노드 1개에만 존재합니다. Tableau Server 클러스터의 각 노드에서 다음 단계를 수행하면서 "independent" 문자열을 찾습니다.

1. EDG 설정의 Tableau Server 노드 1~4에서 tabadminagent 로그 파일 위치를 찾습니다.

   ```
   cd /data/tableau_data/data/tabsvc/logs/tabadminagent
   ```

2. 다음과 같은 이름의 최신 로그를 엽니다.

   ```
   less tabadminagent_nodeN.log
   ```

   (N을 노드 번호로 바꾸기)

3. 다음 검색 문자열을 사용하여 "Independent" 및 "independent"가 나오는 모든 경우를 검색합니다.

   ```
   /ndependent
   ```

   일치하는 항목이 없는 경우 다음 노드로 이동하고 1~3단계를 반복합니다.

4. 일치하는 항목이 발견되면 `Shift + G`를 눌러 맨 아래로 이동하고 마지막 오류 메시지를 확인합니다.

httpd stub 파일 다시 로드

독립 게이트웨이는 Apache의 httpd 구성을 관리합니다. 일시적인 문제를 해결하는 일반적인 작업은 기초 Apache 구성을 채우는 httpd stub 파일을 다시 로드하는 것입니다. 독립 게이트웨이의 두 인스턴스 모두에서 다음 명령을 실행합니다.

1. stub 파일을 httpd.conf에 복사합니다.

```
cp /var/opt/tableau/tableau_tsig/config/httpd.conf.stub
/var/opt/tableau/tableau_tsig/config/httpd.conf
```

2. 독립 게이트웨이 서비스를 다시 시작합니다.

```
sudo su - tableau-tsig
systemctl --user restart tsig-httpd
exit
```

로그 파일 삭제 또는 이동

독립 게이트웨이는 모든 액세스 이벤트를 기록합니다. 디스크 공간이 가득 차는 것을 방지하기 위해 로그 파일 저장소를 관리해야 합니다. 디스크가 가득 차면 독립 게이트웨이가 액세스 이벤트를 기록할 수 없으며 서비스가 실패합니다. 다음 메시지가 독립 게이트웨이의 error.log에 기록됩니다.

```
(28)No space left on device: [client 10.0.2.209:54332] AH00646:
Error writing to /var/opt/tableau/tableau_
tsig/logs/access.%Y_%m_%d_%H_%M_%S.log
```

이 오류는 Tableau 노드 1에서 `tsm status -v`를 실행할 때 `external` 노드에 대해 `DEGRADED` 상태를 초래합니다. 상태 출력의 `external` 노드는 독립 게이트웨이를 참조합니다.

이 문제를 해결하려면 access.log 파일을 삭제하거나 디스크에서 옮기십시오. Access.log 파일은 `/var/opt/tableau/tableau_tsig/logs`에 저장됩니다. 디스크를 지운 후 tableau-tsig 서비스를 다시 시작하십시오.

브라우저 오류

잘못된 요청: 이 시나리오의 일반적인 오류는 Okta의 "잘못된 요청" 오류입니다. 이 문제는 주로 브라우저가 이전 Okta 세션의 데이터를 캐싱할 때 발생합니다. 예를 들어 Okta 관리자 자격으로 Okta 응용 프로그램을 관리하고 다른 Okta 지원 계정을 사용하여 Tableau에 액세스하려고 하면 관리자 데이터의 세션 데이터가 "잘못된 요청" 오류를 발

생시킬 수 있습니다. 로컬 브라우저 캐시를 지워도 이 오류가 계속되면 다른 브라우저로 연결하여 Tableau 시나리오의 유효성을 확인해 보십시오.

"Bad Request" 오류의 또 다른 원인은 Okta, Mellon 및 SAML 구성 프로세스 중에 입력한 많은 URL 중 하나의 입력 오류입니다. 이 모든 것을 오류 없이 입력했는지 확인합니다.

오류를 야기한 URL은 주로 독립 게이트웨이 서버의 `error.log` 파일에 명시됩니다.

찾을 수 없음 - 요청한 URL을 이 서버에서 찾지 못했습니다: 이 오류는 많은 구성 오류 중 하나를 나타냅니다.

사용자가 Okta로 인증된 후 이 오류가 발생하면 SAML을 구성할 때 Okta 사전 인증 응용 프로그램을 Tableau Server에 업로드한 것일 수 있습니다. Okta 사전 인증 응용 프로그램 메타데이터가 아니라 Tableau Server에 Okta Tableau Server 응용 프로그램 메타데이터가 구성되어 있는지 확인합니다.

다른 문제 해결 단계:

- Okta 사전 인증 응용 프로그램 설정을 검토합니다. HTTP와 HTTPS 프로토콜이 이 항목에서 지정한 대로 설정되어 있는지 확인하십시오.
- 두 독립 게이트웨이 서버에서 Restart tsig-httpd를 다시 시작합니다.
- 두 독립 게이트웨이 모두에서 `sudo apachectl configtest`가 "Syntax OK"를 반환하는지 확인합니다.
- 테스트 사용자가 Okta의 두 응용 프로그램에 모두 할당되었는지 확인합니다.
- 부하 분산 장치 및 연결된 대상 그룹에 연결 유지가 설정되어 있는지 확인합니다.

Tableau Server에서 독립 게이트웨이로의 TLS 연결 확인

`wget` 명령을 사용하여 Tableau Server에서 독립 게이트웨이로의 연결 및 액세스를 확인합니다. 이 명령의 변형을 사용하면 인증서 문제가 연결 문제를 일으키는지 여부를 확인하는 데 도움이 될 수 있습니다.

예를 들어 다음 `wget` 명령을 사용하여 Tableau Server의 HK(하우스키핑) 프로토콜을 확인합니다.

```
wget https://ip-10-0-1-38.us-west-1.compute.internal:21319
```

tsig.json 파일의 호스트 옵션에 포함한 것과 동일한 호스트 주소를 사용하여 URL을 구성합니다. `https` 프로토콜을 지정하고 URL에 HK 포트 `21319`를 추가합니다.

연결을 확인하고 인증서 확인을 무시하려면

```
wget https://ip-10-0-1-38.us-west-1.compute.internal:21319 --no-check-certificate
```

TSIG의 루트 CA가 유효한지 확인하려면

```
wget https://ip-10-0-1-38.us-west-1.compute.internal:21319 --ca-certificate=tsigRootCA.pem
```

Tableau에서 통신할 수 있는 경우에는 콘텐츠 관련 오류가 계속해서 발생할 수 있지만 연결 관련 오류는 발생하지 않습니다. Tableau에서 전혀 연결할 수 없는 경우 방화벽/보안 그룹의 프로토콜 구성을 확인하는 것으로 시작합니다. 예를 들어 독립 게이트웨이가 상주하는 보안 그룹의 인바운드 규칙은 TCP 21319를 허용해야 합니다.

부록 - AWS 배포 도구 상자

이 항목에는 AWS에서 배포할 때의 참조 아키텍처에 대한 도구 및 대체 배포 옵션이 포함되어 있습니다. 구체적으로, 이 항목에서는 EDG에 설명된 예제 AWS 배포를 자동화하는 방법을 설명합니다.

TabDeploy4EDG 자동 설치 스크립트

TabDeploy4EDG 스크립트는 4부 - Tableau Server 설치 및 구성에 설명되어 있는 4노드 Tableau 배포의 구현을 자동화합니다. 이 가이드에 설명된 예제 AWS 구현을 따르면 TabDeploy4EDG를 실행할 수 있습니다.

요구 사항. 스크립트를 실행하려면 3부 - Tableau Server 엔터프라이즈 배포 준비의 예제 구현에 따라 AWS 환경을 준비 및 구성해야 합니다.

- VPC, 서브넷 및 보안 그룹이 설명된 대로 구성해야 합니다. IP 주소는 구현 예에 표시된 것과 일치하지 않아도 됩니다.
- 4개의 EC2 인스턴스가 AWS Linux 2의 최신 업데이트 빌드를 실행해야 합니다.
- PostgreSQL이 설치되었으며 PostgreSQL 설치, 구성 및 tar 백업 만들기에 설명된 대로 구성되어 있어야 합니다.
- 1단계 tar 백업 파일은 PostgreSQL 1단계 tar 백업 만들기에 설명된 대로 PostgreSQL이 설치된 EC2 인스턴스에 있습니다.
- Tableau Server 배포의 노드 1을 실행할 EC2 인스턴스는 4부 - Tableau Server 설치 및 구성에 설명된 대로 PostgreSQL과 통신하도록 구성되어 있어야 합니다.
- 배스천 호스트의 SSH 세션을 통해 각 EC2 인스턴스에 로그인되어 있어야 합니다.

이 스크립트가 네 개의 Tableau Server를 설치하고 구성하는 데 약 1.5~2시간이 소요됩니다. 이 스크립트는 참조 아키텍처의 지정된 설정에 따라 Tableau를 구성합니다. 스크립트는 다음 동작을 수행합니다.

- PostgreSQL 호스트의 tar 파일에 대한 경로가 지정된 경우 PostgreSQL 호스트의 1단계 백업을 복원합니다.
- 모든 노드에서 기존 Tableau 설치를 삭제합니다.
- 모든 노드에서 `sudo yum update`를 실행합니다.

- 각 노드에 Tableau rpm 패키지를 다운로드하고 복사합니다.
- 종속 항목을 각 노드에 다운로드하고 설치합니다.
- `/app/tableau_server`를 만들고 모든 노드에 패키지를 설치합니다.
- 로컬 ID 저장소와 함께 노드 1을 설치하고 PostgreSQL을 사용하여 외부 리포지토리를 구성합니다.
- 노드 2 - 노드 4의 부트스트랩 설치와 초기 구성을 수행합니다.
- TabDeploy4EDG의 부트스트랩 파일 및 구성 파일을 삭제합니다.
- 참조 아키텍처 사양에 따라 Tableau 클러스터 전체에 서비스를 구성합니다.
- 설치 유효성을 검사하고 각 노드의 상태를 반환합니다.

배스천 호스트에 스크립트 다운로드 및 복사

1. TabDeploy4EDG 샘플 페이지에서 스크립트를 복사하여 `TabDeploy4EDG`이라는 파일에 코드를 붙여 넣습니다.
2. 배스천 호스트 역할을 하는 EC2 호스트의 홈 디렉터리에 파일을 저장합니다.
3. 다음 명령을 실행하여 파일의 모드를 실행 가능하도록 변경합니다

```
sudo chmod +x TabDeploy4EDG
```

TabDeploy4EDG 실행

배스천 호스트에서 TabDeploy4EDG를 실행해야 합니다. 이 스크립트는 예: AWS의 배스천 호스트에 연결에 설명된 대로 ssh 정방향 에이전트 컨텍스트에서 실행된다는 가정하에 작성되었습니다. ssh 정방향 에이전트 컨텍스트에서 실행하지 않는 경우 설치 프로세스 중에 비밀번호를 입력하라는 메시지가 표시됩니다.

1. 등록 파일(`registration.json`)을 만들고 편집하고 저장합니다. 파일은 올바르게 서식이 지정된 json 파일이어야 합니다. 다음 템플릿을 복사하고 사용자 지정합니다.

```
{
        "zip" : "97403",
        "country" : "USA",
        "city" : "Springfield",
        "last_name" : "Simpson",
        "industry" : "Energy",
        "eula" : "yes",
```

```
        "title" : "Safety Inspection Engineer",
        "phone" : "5558675309",
        "company" : "Example",
        "state" : "OR",
        "department" : "Engineering",
        "first_name" : "Homer",
        "email" : "homer@example.com"
}
```

2. 다음 명령을 실행하여 템플릿 구성 파일을 생성합니다.

   ```
   ./TabDeploy4EDG -g edg.config
   ```

3. 편집할 구성 파일을 엽니다.

   ```
   sudo nano edg.config
   ```

   최소한 각 EC2 호스트의 IP 주소, 등록 파일의 파일 경로 및 유효한 라이선스 키를 추가해야 합니다.

4. 구성 파일 편집이 완료되면 저장한 다음 닫습니다.

5. TabDeploy4EDG를 실행하려면 다음 명령을 실행합니다.

   ```
   ./TabDeploy4EDG -f edg.config
   ```

예: Terraform을 사용하여 AWS 인프라 배포 자동화

이 섹션에서는 Terraform을 구성하고 실행하여 AWS에서 EDG 참조 아키텍처를 배포하는 방법을 설명합니다. 여기에 나온 예제 Terraform 구성은 3부 - Tableau Server 엔터프라이즈 배포 준비에 설명된 서브넷, 보안 그룹 및 EC2 인스턴스가 포함되어 있는 AWS VPC를 배포합니다.

샘플 Terraform 템플릿은 Tableau Samples 웹 사이트 (https://help.tableau.com/samples/en-us/edg/edg-terraform.zip)에서 사용할 수 있습니다.

해당 조직에 맞게 이 템플릿을 구성하고 사용자 지정해야 합니다. 이 섹션에 제공된 구성 콘텐츠는 배포를 위해 사용자 지정해야 하는 최소 템플릿 변경 사항에 대해 설명합니다.

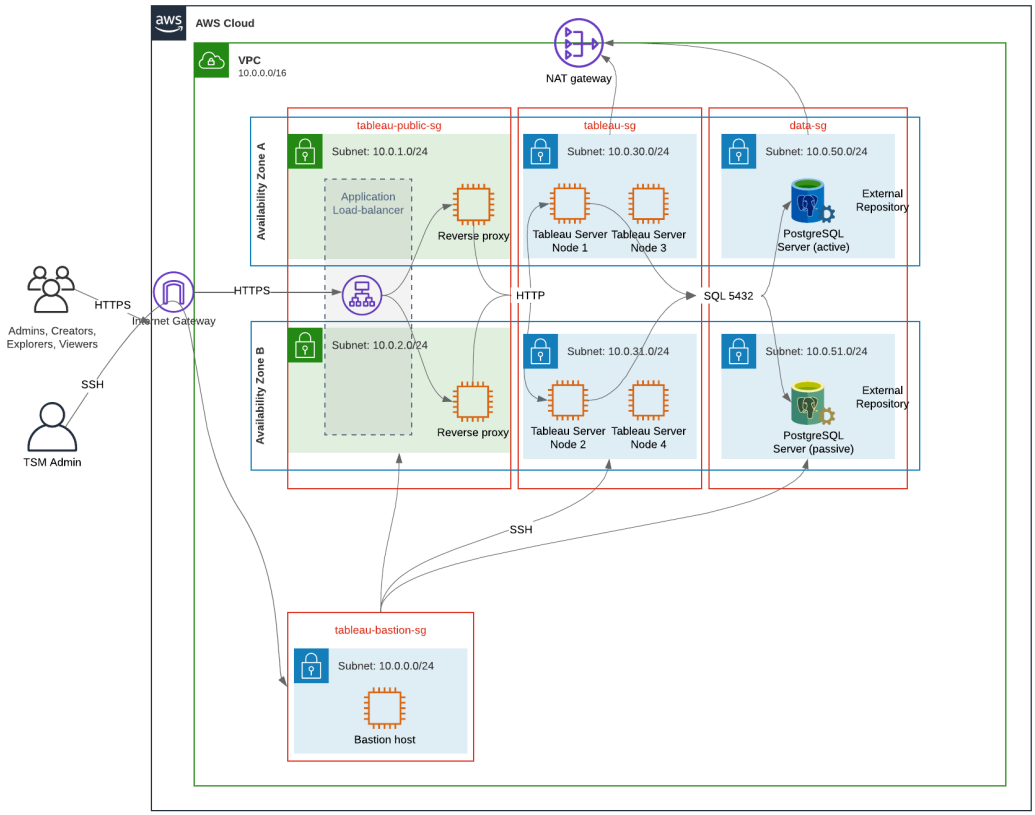
목표

여기에 제공된 Terraform 템플릿 및 콘텐츠는 개발 테스트 환경에 EDG를 빠르게 배포할 수 있는 작업 샘플을 제공하기 위한 것입니다.

Tableau는 예제 Terraform 배포를 테스트하고 문서화하는 데 최선의 노력을 기울였습니다. 그러나 프로덕션 환경에서 Terraform을 사용하여 EDG를 배포하고 유지 관리하려면 이 예제의 범위를 벗어나는 Terraform 전문 지식이 필요합니다. Tableau는 여기에 문서화되어 있는 예제 Terraform 솔루션에 대한 지원을 제공하지 않습니다.

최종 상태

이 섹션의 절차에 따라 AWS에서 3부 - Tableau Server 엔터프라이즈 배포 준비에 명시된 VPC와 기능적으로 동일한 VPC를 설정합니다.



이 섹션의 샘플 Terraform 템플릿 및 지원 콘텐츠:

- 탄력적 IP 주소, 가용성 영역 2개 및 위에 표시된 서브넷 조직(IP 주소는 다름)을 사용하여 VPC를 만듭니다.
- 배스천, 공용, 사설 및 데이터 보안 그룹을 만듭니다.
- 보안 그룹에 대한 대부분의 수신 및 송신 규칙을 설정합니다. Terraform이 실행된 후 보안 그룹을 편집해야 합니다.
- 다음과 같은 EC2 호스트(각각 AWS Linux2를 실행)를 만듭니다. 배스천, 프록시 1, 프록시 2, Tableau 노드 1, Tableau 노드 2, Tableau 노드 3, Tableau 노드 4
- PostgreSQL용 EC2 호스트는 만들어지지 않습니다. 데이터 보안 그룹에 EC2를 수동으로 만든 다음 PostgreSQL 설치, 구성 및 tar 백업 만들기에 설명된 대로 PostgreSQL을 설치하고 구성해야 합니다.

요구 사항

- AWS 계정 - VPC를 만들 수 있는 AWS 계정에 액세스할 수 있어야 합니다.
- Windows 컴퓨터에서 Terraform을 실행하는 경우 AWS CLI를 설치해야 합니다.
- AWS 계정에서 사용할 수 있는 탄력적 IP 주소
- AWS Route 53에 등록된 도메인 Terraform은 Route 53에 DNS 영역과 관련 SSL 인증서를 만듭니다. 따라서 Terraform을 실행하는 프로필 또한 Route 53에서 적절한 사용 권한을 가지고 있어야 합니다.

시작하기 전에

- 이 절차의 명령줄 예제는 Apple OS 사용 터미널에 대한 것입니다. Windows에서 Terraform을 실행하는 경우 파일 경로를 사용하여 적절하게 명령을 조정해야 합니다.
- Terraform 프로젝트는 다수의 텍스트 구성 파일로 구성됩니다(.tf 파일 확장자). 이러한 파일을 사용자 지정하여 Terraform을 구성합니다. 강력한 텍스트 에디터가 없는 경우 Atom 또는 Text++를 설치합니다.
- Terraform 프로젝트를 다른 사용자와 공유하는 경우 변경 관리를 위해 프로젝트를 Git에 저장하는 것이 좋습니다.

1단계 - 환경 준비

A. Terraform 다운로드 및 설치

https://www.terraform.io/downloads

B. 공용-개인 키 쌍 생성

AWS와 결과 VPC 환경에 액세스할 때 사용할 키입니다. Terraform을 실행할 때 공용 키를 포함합니다.

터미널을 열고 다음 명령을 실행합니다.

1. Create a private key. For example, `my-key.pem`:

   ```
   openssl genrsa -out my-key.pem 1024
   ```

2. 공용 키를 만듭니다. 이 키 형식은 Terraform에 사용되지 않습니다. 이 절차의 나중 부분에서 ssh 키로 변환합니다.

   ```
   openssl rsa -in my-key.pem -pubout > my-key.pub
   ```

3. 개인 키에 대한 사용 권한을 설정합니다.

   ```
   sudo chmod 0600 my-key.pem
   ```

   Windows에서 사용 권한을 설정하려면:

   - Windows Explorer에서 파일을 찾으려면 오른쪽 클릭한 다음 **속성**을 선택합니다. **보안** 탭으로 이동한 다음 **고급**을 클릭합니다.
   - 소유자를 자신으로 변경하고 상속을 사용하지 않도록 설정한 다음 모든 사용 권한을 삭제합니다. 자신에게 **모든 권한**을 부여한 다음 **저장**을 클릭합니다. 파일을 읽기 전용으로 표시합니다.

4. ssh 공용 키를 만듭니다. 프로세스의 나중 부분에서 이 키를 Terraform에 복사합니다.

   ```
   ssh-keygen -y -f my-key.pem >my-key-ssh.pub
   ```

C. 프로젝트 다운로드 및 상태 디렉터리 추가

1. EDG Terraform 프로젝트를 다운로드하고 압축을 푼 다음 로컬 컴퓨터에 저장합니다. 다운로드 압축을 풀면 최상위 디렉터리, edg-terraform 및 일련의 하위 디렉터리가 표시됩니다.

2. `state`를 디렉터리를 최상위 `edg-terraform` 디렉터리의 피어로 만듭니다.

2단계: Terraform 템플릿 사용자 지정

해당하는 AWS 및 EDG 환경에 맞게 Terraform 템플릿을 사용자 지정해야 합니다. 여기의 예제에는 대부분의 조직에서 수행해야 하는 최소 템플릿 사용자 지정이 나와 있습니다. 사용하는 특정 환경에 따라 다른 사용자 지정이 필요할 가능성이 높습니다.

이 섹션은 템플릿 이름으로 구성됩니다.

*3단계 - Terraform 실행*으로 계속하기 전에 모든 변경 내용을 저장하십시오.

versions.tf

There are three instances of `versions.tf` files where the `required_version` field must match the version of terraform.exe you're using. Check the version of terraform (`terraform.exe -version`) and update each of the following instances:

- `edg-terraform\versions.tf`
- `edg-terraform\modules\proxy\versions.tf`
- `edg-terraform\modules\tableau_instance\versions.tf`

key-pair.tf

1. 1B단계에서 생성한 공용 키를 열고 키를 복사합니다.

   ```
   less my-key-ssh.pub
   ```

   Windows: 공용 키의 내용을 복사합니다.

2. 공용 키 문자열을 public_key 인수에 복사합니다. 예를 들면 다음과 같습니다.

   ```
   resource "aws_key_pair" "tableau" {
   key_name = "my-key"
   public_key = "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAAAgQ (truncated
   example) dZVHambOCw=="
   ```

Ensure that the `key_name` value is unique in the datacenter or `terraform apply` will fail.

locals.tf

Update `user.owner` to your name or alias. The value you enter here will be used for the "Name" tag in AWS on the resources that Terraform creates.

providers.tf

1. 조직의 요구 사항에 따라 태그를 추가합니다. 예를 들면 다음과 같습니다.

   ```
   default_tags {
     tags = {
   ```

```
    "Application" = "tableau",
    "Creator" = "alias@example.com",
    "DeptCode" = "8675309",
    "Description" = "EDG",
    "Environment" = "test",
    "Group" = "itcloud@example.com"
   }
}
```

2. If using `provider`, comment out the `assume_role` lines:

```
/* assume_role {
role_arn     = "arn:aws:iam::310946706895:role/terraform-
backend"
session_name = "terraform"
}*/
```

elb.tf

Under `'resource "aws_lb" "tableau" {'` choose a unique value to use for `name` and `tags.Name`.

If another AWS load balancer has the same name in the datacenter, then `terraform apply` will fail.

Add `idle_timeout`:

```
resource "aws_lb" "tableau" {
name                       = "edg-again-alb"
load_balancer_type         = "application"
subnets                    = [for subnet in aws_subnet.public :
subnet.id]
security_groups            = [aws_security_group.public.id]
drop_invalid_header_fields = true
idle_timeout               = 400
tags = {
Name = "edg-again-alb"
```

```
}
                                    }
```

variables.tf

루트 도메인 이름을 업데이트합니다. 이 이름은 Route 53에서 등록한 도메인과 일치해야 합니다.

```
variable "root_domain_name" {
 default = "example.com"
 }
```

기본적으로 `tableau`라는 하위 도메인이 VPC DNS 도메인 이름에 지정됩니다. 이를 변경하려면 `subdomain`을 업데이트합니다.

```
variable "subdomain" {
 default = "tableau"
}
```

modules/tableau_instance/ec2.tf

There are two ec2.tf files in the project. This customization is for the Tableau instance of the ec2.tf in the directory: `modules/tableau_instance/ec2.tf`.

- 필요한 경우 태그 blob을 추가합니다.

```
tags = {
 "Name" : var.ec2_name,
 "user.owner" = "ALIAS",
 "Application" = "tableau",
 "Creator" = "ALIAS@example.com",
 "DeptCode" = "8675309",
 "Description" = "EDG",
 "Environment" = "test",
 "Group" = "itcloud@example.com"
 }
}
```

- 필요에 따라 데이터 요구 사항을 처리하도록 저장소를 업데이트합니다.

  루트 볼륨:

  ```
  root_block_device {
   volume_size = 100
   volume_type = "gp3"
  }
  ```

  응용 프로그램 볼륨:

  ```
  resource "aws_ebs_volume" "tableau" {
   availability_zone = data.aws_subnet.tableau.availability_zone
   size              = 500
   type              = "gp3"
  }
  ```

3단계: Terraform 실행

A. Terraform 초기화

터미널에서 `edg-terraform` 디렉터리로 변경하고 다음 명령을 실행합니다.

```
terraform init
```

초기화에 성공하면 다음 단계로 계속합니다. 초기화에 실패하면 Terraform 출력의 지침을 따릅니다.

B. Terraform 계획

동일한 디렉터리에서 plan 명령을 실행합니다.

```
terraform plan
```

이 명령을 여러 번 실행해도 됩니다. 오류를 수정하는 데 필요한 만큼 여러 번 실행합니다. 이 명령이 오류 없이 실행되면 다음 단계로 계속합니다.

C. Terraform 적용

동일한 디렉터리에서 apply 명령을 실행합니다.

```
terraform apply
```

Terraform will prompt you to verify deployment, type `Yes`.

선택 사항: Terraform 삭제

destroy 명령을 실행하여 전체 VPC를 삭제할 수 있습니다.

```
terraform destroy
```

destroy 명령은 이전에 만든 것만 삭제합니다. AWS에서 일부 개체(보안 그룹, 서브넷 등)를 수동으로 변경한 경우 `destroy`가 실패합니다. destroy 작업이 실패/중단된 경우 종료하려면 `Control + C`를 입력합니다. 그런 다음 VPC를 Terraform에서 원래 만들 때의 상태로 수동으로 정리해야 합니다. 그런 다음 `destroy` 명령을 실행할 수 있습니다.

4단계 - 배스천에 연결

모든 명령줄 연결은 TCP 22(SSH 프로토콜)의 배스천 호스트를 통합니다.

1. AWS에서 배스천 보안 그룹을 만들고(**AWS > 보안 그룹 > 배스천 SG > 인바운드 규칙 편집**) 터미널 명령을 실행하는 IP 주소 또는 서브넷 마스크의 SSH(TCP 22) 연결을 허용하는 규칙을 만듭니다.

   선택 사항: 배포 중에 사설 및 공용 그룹의 EC2 인스턴스 간 파일 복사를 허용하는 것이 좋을 수 있습니다. 인바운드 SSH 규칙 만들기:

   - 사설: 공용의 SSH를 허용하는 인바운드 규칙 만들기
   - 공용: 사설 및 공용의 SSH를 허용하는 인바운드 규칙 만들기

2. 1.B단계에서 만든 pem 키를 사용하여 배스천 호스트에 연결합니다.

   Mac 터미널:

   pem 키가 저장된 디렉터리에서 다음 명령을 실행합니다.

```
ssh-add —apple-use-keychain <keyName>.pem
```

다른 사람이 개인 키에 액세스할 수 있다는 경고가 표시되면 `chmod 600 <keyName.pem>` 명령을 실행한 다음 `ssh-add` 명령을 다시 실행합니다.

`ssh -A ec2-user@IPaddress` 명령을 사용하여 배스천 호스트에 연결합니다.

예를 들어 `ssh -A ec2-user@3.15.12.112`를 사용합니다.

Windows에서 PuTTY 및 Pageant 사용:

a. pem 키에서 ppk 만들기: PuTTY 키 생성기를 사용합니다. 1.B 단계에서 만든 pem 키를 로드합니다. 키 가져오기 후에 **Save private key**(개인 키 저장)를 클릭합니다. 그러면 ppk 파일이 만들어집니다.

b. PuTTY에서 - 구성을 열고 다음과 같이 변경합니다.

   - Sessions>Host Name: 배스천 호스트의 IP 주소를 추가합니다.
   - Sessions>Port: 22
   - Connection>Data>Auto-login username: ec2-user
   - Connection>SSH>Auth>Allow agent forwarding
   - Connection>SSH>Auth> 개인 키의 경우 찾아보기를 클릭하고 방금 만든 .ppk 파일을 선택합니다.

c. Pageant를 설치하고 ppk를 응용 프로그램에 로드합니다.

5단계 - PostgreSQL 설치

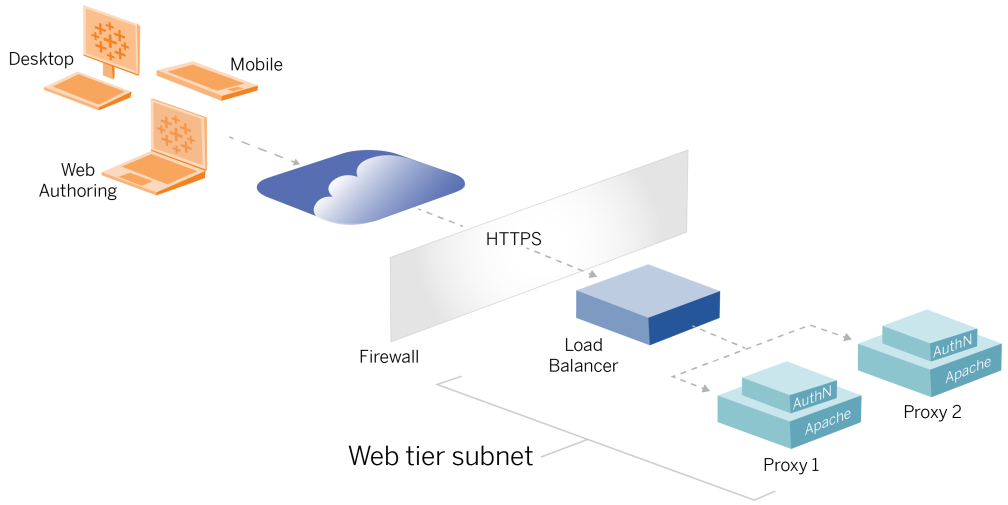
Terraform 템플릿은 외부 리포지토리로 사용할 PostgreSQL을 설치하지 않습니다. 그러나 관련된 보안 그룹과 서브넷은 만들어집니다. PostgreSQL을 실행하는 EC2에 외부 리포지토리를 설치할 예정인 경우 3부 - Tableau Server 엔터프라이즈 배포 준비에 설명된 대로 EC2 인스턴스를 배포해야 합니다.

그런 다음 4부 - Tableau Server 설치 및 구성에 설명된 대로 PostgreSQL을 설치 및 구성하고 tar 백업을 만듭니다.

6단계 - (선택 사항) DeployTab4EDG 실행

TabDeploy4EDG 스크립트는 4부에 설명된 4노드 Tableau 배포의 구현을 자동화합니다. TabDeploy4EDG 자동 설치 스크립트를 참조하십시오.

부록 - Apache를 사용한 웹 계층 예제 배포



이 항목에서는 예제 AWS 참조 아키텍처에서 웹 계층을 구현하는 방법을 설명하는 전체 절차를 제공합니다. 예제 구성은 다음 구성 요소로 구성됩니다.

- AWS 응용 프로그램 부하 분산 장치
- Apache 프록시 서버
- Mellon 인증 모듈
- Okta IdP
- SAML 인증

참고: 이 섹션에 나와 있는 웹 계층 구성 예제에는 타사 소프트웨어 및 서비스를 배포하는 자세한 절차가 포함되어 있습니다. Tableau는 웹 계층 시나리오를 지원하기 위한 절차를 확인하고 문서화하기 위해 최선을 다하고 있습니다. 하지만 타사 소프트웨어가 변경되거나 여기에 설명된 참조 아키텍처와 다른 시나리오가 있을 수 있습니다. 신뢰할 수 있는 구성 세부 정보 및 지원은 타사 설명서를 참조하십시오.

이 섹션 전반에 걸친 Linux 예는 RHEL와 같은 배포에 대한 명령을 보여줍니다. 특히 여기에 나온 명령은 Amazon Linux 2 배포를 통해 개발되었습니다. Ubuntu 배포를 실행 중인 경우 그에 따라 명령을 편집합니다.

이 예제의 웹 계층 배포는 단계별 구성 및 확인 절차를 따릅니다. 핵심 웹 계층 구성은 Tableau와 인터넷 간의 HTTP를 사용하도록 설정하는 다음 단계로 구성됩니다. AWS 응용 프로그램 부하 분산 장치 뒤에서 역방향 프록시/부하 분산을 수행하도록 Apache를 구성하고 실행합니다.

1. Apache 설치
2. Tableau Server에 대한 연결을 테스트하도록 역방향 프록시 구성
3. 프록시의 부하 분산 구성
4. AWS 응용 프로그램 부하 분산 장치 구성

웹 계층을 설정하고 Tableau 연결을 확인한 후에는 외부 공급자를 통한 인증을 구성합니다.

Apache 설치

두 EC2 호스트(프록시 1 및 프록시 2)에서 다음 절차를 실행합니다. 참조 아키텍처 예제에 따라 AWS에서 배포하는 경우 두 가용성 영역이 있어야 하며 각 영역에서 단일 프록시 서버를 실행해야 합니다.

1. Apache를 설치합니다.

   ```
   sudo yum update -y
   sudo yum install -y httpd
   ```

2. 재부팅 시 Apache를 시작하도록 구성합니다.

   ```
   sudo systemctl enable --now httpd
   ```

3. 설치한 httpd 버전에 `proxy_hcheck_module`이 있는지 확인합니다.

   ```
   sudo httpd -M
   ```

proxy_hcheck_module이 필요합니다. httpd 버전에 이 모듈이 없는 경우 모듈이 포함된 httpd 버전으로 업데이트합니다.

Tableau Server에 대한 연결을 테스트하도록 프록시 구성

프록시 호스트 중 하나(프록시 1)에서 이 절차를 실행합니다. 이 단계의 목적은 사설 보안 그룹에서 인터넷, 프록시 서버 및 Tableau Server 간의 연결을 확인하는 것입니다.

1. tableau.conf 파일을 만들고 /etc/httpd/conf.d 디렉터리에 추가합니다.

   다음 코드를 복사하고 Tableau Server 노드 1의 비공개 IP 주소를 사용하여 ProxyPass 및 ProxyPassReverse 키를 지정합니다.

   > **중요**: 아래에 표시된 구성은 안전하지 않으므로 프로덕션에서 사용되어서는 안 됩니다. 이 구성은 설치 프로세스 중에 종단간 연결을 확인하기 위한 용도로만 사용되어야 합니다.

   예를 들어 노드 1의 사설 IP 주소가 10.0.30.32인 경우 tableau.conf 파일의 내용은 다음과 같습니다.

   ```
   <VirtualHost *:80>
   ProxyPreserveHost On
   ProxyPass "/" "http://10.0.30.32:80/"
   ProxyPassReverse "/" "http://10.0.30.32:80/"
   </VirtualHost>
   ```

2. httpd를 다시 시작합니다.

   ```
   sudo systemctl restart httpd
   ```

확인: 기본 토폴로지 구성

`http://<proxy-public-IP-address>`로 이동하여 Tableau Server 관리 페이지에 액세스할 수 있어야 합니다.

브라우저에서 Tableau Server 로그인 페이지가 로드되지 않는 경우 프록시 1 호스트에서 다음 문제 해결 단계를 수행하십시오.

- 첫 번째 문제 해결 단계로 httpd를 중지한 다음 시작합니다.
- `tableau.conf` 파일을 다시 확인합니다. 노드 1 비공개 IP가 올바른지 확인합니다. 큰따옴표를 확인하고 구문을 세심히 검토합니다.
- 역방향 프록시 서버에서 노드 1 비공개 IP 주소를 사용하여 `curl` 명령을 실행합니다(예: `curl 10.0.1.90`). 셸에서 html이 반환되지 않거나 Apache 테스트 웹 페이지의 html이 반환되는 경우 공용 및 사설 보안 그룹 간의 프로토콜/포트 구성을 확인합니다.
- 프록시 1 비공개 IP 주소를 사용하여 `curl` 명령을 실행합니다(예: `curl 10.0.0.163`). 셸에서 Apache 테스트 웹 페이지에 대한 html 코드가 반환되는 경우 프록시 파일이 올바르게 구성되지 않은 것입니다.
- 프록시 파일 또는 보안 그룹의 구성을 변경한 후에는 항상 httpd(`sudo systemctl restart httpd`)를 다시 시작합니다.
- TSM이 노드 1에서 실행되고 있는지 확인합니다.

프록시의 부하 분산 구성

1. `tableau.conf` 파일을 만든 동일한 프록시 호스트(프록시 1)에서 기존 가상 호스트 구성을 제거하고 부하 분산 논리를 포함하도록 파일을 편집합니다.

   예:

   ```
   <VirtualHost *:80>
   ServerAdmin admin@example.com
   #Load balancing logic.
   ProxyHCExpr ok234 {%{REQUEST_STATUS} =~ /^[234]/}
   Header add Set-Cookie "ROUTEID=.%{BALANCER_WORKER_ROUTE}e;
   path=/" env=BALANCER_ROUTE_CHANGED
   <Proxy balancer://tableau>
   ```

```
#Replace IP addresses below with the IP addresses to the
Tableau Servers running the Gateway service.
BalancerMember http://10.0.3.40/ route=1 hcmethod=GET
hcexpr=ok234 hcuri=/favicon.ico
BalancerMember http://10.0.4.151/ route=2 hcmethod=GET
hcexpr=ok234 hcuri=/favicon.ico
ProxySet stickysession=ROUTEID
</Proxy>
ProxyPreserveHost On
ProxyPass / balancer://tableau/
ProxyPassReverse / balancer://tableau/
</VirtualHost>
```

2. httpd를 중지한 다음 시작합니다.

```
sudo systemctl stop httpd
sudo systemctl start httpd
```

3. 프록시 1의 공용 IP 주소로 이동하여 구성을 확인합니다.

두 번째 프록시 서버에 구성 복사

1. 프록시 1에서 `tableau.conf` 파일을 복사하고 프록시 2 호스트의 `/etc/httpd/conf.d` 디렉터리에 저장합니다.

2. httpd를 중지한 다음 시작합니다.

```
sudo systemctl stop httpd
sudo systemctl start httpd
```

3. 프록시 2의 공용 IP 주소로 이동하여 구성을 확인합니다.

AWS 응용 프로그램 부하 분산 장치 구성

부하 분산 장치를 HTTP 수신기로 구성합니다. 다음 절차에서는 AWS에서 부하 분산 장치를 추가하는 방법에 대해 설명합니다.

1단계: 대상 그룹 만들기

대상 그룹은 프록시 서버를 실행하는 EC2 인스턴스를 정의하는 AWS 구성입니다. 이러한 그룹은 LBS에서 오는 트래픽의 대상입니다.

1. EC2 > **대상 그룹** > **대상 그룹 만들기**

2. 만들기 페이지에서:

   - 대상 그룹 이름 입력(예: `TG-internal-HTTP`)
   - 대상 유형: 인스턴스
   - 프로토콜: HTTP
   - 포트: 80
   - VPC: 해당하는 VPC 선택
   - **상태 확인** > **고급 상태 확인 설정** > **성공 코드**에서 읽을 코드 목록(`200,303`)을 추가합니다.
   - **만들기**를 클릭합니다.

3. 방금 만든 대상 그룹을 선택한 다음 **대상** 탭을 클릭합니다.

   - **편집**을 클릭합니다.
   - 프록시 애플리케이션을 실행하는 EC2 인스턴스(또는 한 번에 1개를 구성하는 경우 단일 인스턴스)를 선택한 다음 **등록된 항목에 추가**를 클릭합니다.
   - **저장**을 클릭합니다.

2단계: 부하 분산 장치 마법사 시작

1. EC2 > **부하 분산 장치** > **부하 분산 장치 만들기**

2. "부하 분산 장치 유형 선택" 페이지에서 응용 프로그램 부하 분산 장치를 만듭니다.

> **참고**: 부하 분산 장치를 구성하기 위해 표시되는 UI는 AWS 데이터 센터 간에 일관되지 않습니다. 아래의 "마법사 구성" 절차는 **1단계 부하 분산 장치 구성**으로 시작하는 AWS 구성 마법사를 설명합니다.
>
> 데이터 센터가 모든 구성을 단일 페이지에 표시하는 경우 페이지 아래쪽에 **부하 분**

산 장치 만들기 단추가 포함되어 있으면 아래의 "단일 페이지 구성" 절차를 따르십시오.

마법사 구성

1. **부하 분산 장치 구성** 페이지:
   - 이름 지정
   - 스키마: 인터넷 연결(기본값)
   - IP 주소 유형: ipv4(기본값)
   - 수신기(수신기 및 라우팅):
     a. 기본 HTTP 수신기를 그대로 둡니다.
     b. **수신기 추가**를 클릭하고 `HTTPS:443`를 추가합니다.
   - VPC: 모든 항목을 설치한 VPC 선택
   - 가용성 영역:
     - 데이터 센터 지역을 나타내는 **a** 및 **b**를 선택합니다.
     - 해당하는 각 드롭다운 선택 도구에서 공용 서브넷(프록시 서버가 상주하는 서브넷)을 선택합니다.
   - **보안 설정 구성** 클릭

2. **보안 설정 구성** 페이지
   - 공용 SSL 인증서를 업로드합니다.
   - **다음: 보안 그룹 구성**을 클릭합니다.

3. **보안 그룹 구성** 페이지:
   - 공용 보안 그룹을 선택합니다. 기본 보안 그룹이 선택된 경우 해당 선택을 취소합니다.
   - **다음: 라우팅 구성**을 클릭합니다.

4. **라우팅 구성** 페이지
   - 대상 그룹: 기존 대상 그룹
   - 이름: 이전에 만든 대상 그룹 선택
   - **다음: 대상 등록**을 클릭합니다.

5. **대상 등록** 페이지

- 이전에 구성한 프록시 서버 인스턴스 2개가 표시됩니다.
- **다음: 검토**를 클릭합니다.

6. **검토** 페이지

   만들기를 클릭합니다.

단일 페이지 구성

기본 구성

- 이름 지정
- 스키마: 인터넷 연결(기본값)
- IP 주소 유형: ipv4(기본값)

네트워크 매핑

- VPC: 모든 항목을 설치한 VPC 선택
- 매핑:
  - 데이터 센터 지역을 나타내는 **a** 및 **b**(또는 그에 상응하는) 가용성 영역을 선택합니다.
  - 해당하는 각 드롭다운 선택 도구에서 공용 서브넷(프록시 서버가 상주하는 서브넷)을 선택합니다.

보안 그룹

공용 보안 그룹을 선택합니다. 기본 보안 그룹이 선택된 경우 해당 선택을 취소합니다.

수신기 및 라우팅

- 기본 HTTP 수신기를 그대로 둡니다. **기본 동작**의 경우 이전에 설정한 대상 그룹을 지정합니다.
- **수신기 추가**를 클릭하고 `HTTPS:443`을 추가합니다. **기본 동작**의 경우 이전에 설정한 대상 그룹을 지정합니다.

보안 수신기 설정

- 공용 SSL 인증서를 업로드합니다.

부하 분산 장치 만들기를 클릭합니다.

3단계: 연결 유지 사용

1. 부하 분산 장치를 만든 후 대상 그룹에서 연결 유지를 사용하도록 설정해야 합니다.
    - AWS 대상 그룹 페이지(**EC2 > 부하 분산 > 대상 그룹**)를 열고 방금 설정한 대상 그룹 인스턴스를 선택합니다. **동작** 메뉴에서 **특성 편집**을 선택합니다.
    - **특성 편집** 페이지에서 **연결 유지**를 선택하고 기간을 `1 day`로 지정한 다음 **변경 내용을 저장**합니다.
2. 부하 분산 장치에서 HTTP 수신기에 연결 유지를 사용하도록 설정합니다. 방금 구성한 부하 분산 장치를 선택한 다음 **수신기** 탭을 클릭합니다.
    - **HTTP:80**에서 **규칙 보기/편집**을 클릭합니다. 결과로 표시되는 **규칙** 페이지에서 편집 아이콘을 클릭하여 규칙을 편집합니다(페이지 맨 위에서 한 번 클릭하여 편집한 다음 규칙별로 다시 편집). 기존 THEN 규칙을 삭제하고 **동작 추가 > 전달 대상...**을 클릭하여 바꿉니다. 결과로 나온 THEN 구성에서 이전에 만든 것과 동일한 대상 그룹을 지정합니다. 그룹 수준 연결 유지에서 연결 유지를 사용하도록 설정하고 기간을 1일로 설정합니다. 설정을 저장한 다음 **업데이트**를 클릭합니다.

4단계: 부하 분산 장치에서 유휴 시간 초과 설정

부하 분산 장치에서 유휴 시간 초과를 400초로 업데이트합니다.

이 배포에 대해 구성한 부하 분산 장치를 선택한 다음 **동작 > 특성 편집**을 클릭합니다. **유휴 시간 초과**를 `400` 초로 설정한 다음 **저장**을 클릭합니다.

5단계: LBS 연결 확인

AWS 부하 분산 장치 페이지(**EC2 > 부하 분산 장치**)를 열고 방금 설정한 부하 분산 장치 인스턴스를 선택합니다.

설명에서 DNS 이름을 복사하여 브라우저에 붙여 넣고 Tableau Server 로그인 페이지에 액세스합니다.

500 수준 오류가 발생하면 프록시 서버를 다시 시작해야 할 수 있습니다.

공용 Tableau URL로 DNS 업데이트

AWS 부하 분산 장치 설명의 도메인 DNS 영역 이름을 사용하여 DNS에 CNAME 값을 만듭니다. URL(tableau.example.com)에 대한 트래픽은 AWS 공용 DNS 이름으로 전송되어야 합니다.

연결 확인

DNS 업데이트가 완료되면 공용 URL(예: `https://tableau.example.com`)을 입력하여 Tableau Server 로그인 페이지로 이동할 수 있어야 합니다.

인증 구성 예: SAML 및 외부 IdP

다음 예에서는 AWS 참조 아키텍처에서 실행되는 Tableau 배포를 위해 Okta IdP 및 Mellon 인증 모듈로 SAML을 설정하고 구성하는 방법에 대해 설명합니다. 이 예에서는 HTTP를 사용하도록 Tableau Server 및 Apache 프록시 서버를 구성하는 방법을 설명합니다. Okta는 HTTPS를 통해 AWS 부하 분산 장치로 요청을 보내지만 모든 내부 트래픽은 HTTP를 통해 이동합니다. 이 시나리오에서 구성할 때는 URL 문자열을 설정할 때 HTTP와 HTTPS 프로토콜의 차이점을 숙지하십시오.

이 예에서는 Mellon을 역방향 프록시 서버의 사전 인증 서비스 공급자 모듈로 사용합니다. 이 구성에서는 인증된 트래픽만 Tableau Server에 연결되며 Tableau Server는 Okta IdP를 통한 서비스 공급자 역할도 합니다. 따라서 Mellon 서비스 공급자용과 Tableau 서비스 공급자용으로 IdP 응용 프로그램 2개를 구성해야 합니다.

Tableau 관리자 계정 만들기

SAML을 구성할 때 흔히 하는 실수는 SSO를 사용하도록 설정하기 전에 Tableau Server에서 관리자 계정을 생성하는 것을 잊는 것입니다.

첫 번째 단계는 Tableau Server에서 서버 관리자 역할의 계정을 만드는 것입니다. Okta 시나리오의 예에서 사용자 이름은 올바른 이메일 주소 형식이어야 합니다(예:

user@example.com). 이 사용자에 대한 비밀번호를 설정해야 하지만 SAML이 구성된 후에는 비밀번호가 사용되지 않습니다.

Okta 사전 인증 응용 프로그램 구성

이 섹션에 설명된 전체 시나리오에서는 두 개의 Okta 응용 프로그램이 필요합니다.

- Okta 사전 인증 응용 프로그램
- Okta Tableau Server 응용 프로그램

이러한 각 응용 프로그램은 서로 다른 메타데이터에 연결되며 이를 역방향 프록시와 Tableau Server에서 각각 구성해야 합니다.

이 절차에서는 Okta 사전 인증 응용 프로그램을 만들고 구성하는 방법에 대해 설명합니다. 이 항목의 뒷부분에서는 Okta Tableau Server 응용 프로그램을 만듭니다. 사용자 수가 제한되어 있는 Okta 무료 테스트 계정은 Okta 개발자 웹 페이지를 참조하십시오.

Mellon 사전 인증 서비스 공급자를 위한 SAML 앱 통합을 만듭니다.

1. Okta 관리 대시보드 > **응용 프로그램** > **앱 통합 만들기**를 엽니다.
2. **새 앱 통합 만들기** 페이지에서 **SAML 2.0**을 선택한 후 **다음**을 클릭합니다.
3. **일반 설정** 탭에서 앱 이름(예: `Tableau Pre-Auth`)을 입력하고 **다음**을 클릭합니다.
4. **SAML 구성** 탭에서:
   - SSO(Single Sign-On) URL. Single Sign-On URL의 최종 경로 요소는 이 절차의 뒷부분에서 설명하는 `mellon.conf` 구성 파일의 `MellonEndpointPath`를 일컫습니다. 원하는 끝점을 지정할 수 있습니다. 이 예에서는 `sso`가 끝점입니다. 마지막 요소인 `postResponse`는 `https://tableau.example.com/sso/postResponse`에 필수입니다.
   - **수신자 URL 및 대상 URL에 사용** 확인란을 선택 취소합니다.
   - 수신자 URL: SSO URL과 동일하지만 HTTP가 포함됩니다. 예를 들어 `http://tableau.example.com/sso/postResponse`입니다.

- 대상 URL: SSO URL과 동일하지만 HTTP가 포함됩니다. 예를 들어 `http://tableau.example.com/sso/postResponse`입니다.
- 대상 URI(SP 엔터티 ID) 예를 들어 `https://tableau.example.com`입니다.
- 이름 ID 형식: `EmailAddress`
- 응용 프로그램 사용자 이름: `Email`
- 특성 문: 이름 = `mail`, 이름 형식 = `Unspecified`, 값 = `user.email`.

다음을 클릭합니다.

5. **피드백** 탭에서 다음을 선택합니다.
   - **내부 앱을 추가하는 Okta 고객**
   - **이전에 만든 내부 앱**
   - **마침**을 클릭합니다.
6. 사전 인증 IdP 메타데이터 파일을 만듭니다.
   - Okta에서: **Applications(응용 프로그램)** > **Applications(응용 프로그램)** > Your new application(새 응용 프로그램)(예: `Tableau Pre-Auth`) > **Sign On (로그온)**
   - **SAML Signing Certificates(SAML 서명 인증서)** 근처에서 **View SAML setup instructions(SAML 설정 지침 보기)**를 클릭합니다.
   - **How to Configure SAML 2.0 for <pre-auth> Application(<pre-auth> 응용 프로그램에 대해 SAML 2.0을 구성하는 방법)** 페이지에서 **Optional(선택 사항)** 섹션인 **Provide the following IDP metadata to your SP provider(SP 공급자에게 다음 IdP 메타데이터 제공)**로 스크롤합니다.
   - XML 필드의 콘텐츠를 복사하고 `pre-auth_idp_metadata.xml`이라는 파일에 저장합니다.
7. (선택 사항) 다단계 인증을 구성합니다.
   - Okta에서: **Applications(응용 프로그램)** > **Applications(응용 프로그램)** > Your new application(새 응용 프로그램)(예: `Tableau Pre-Auth`) > **Sign On (로그온)**
   - **로그온 정책**에서 **규칙 추가**를 클릭합니다.
   - **앱 로그온 규칙**에서 이름 및 다른 MFA 옵션을 지정합니다. 기능을 테스트하려면 모든 옵션을 기본값으로 두어도 됩니다. 그러나 **동작**에서는 **인증 요**

소 표시를 선택하고 사용자가 로그인해야 하는 빈도를 지정해야 합니다. **저장**을 클릭합니다.

Okta 사용자 만들기 및 할당

1. Okta에서 **디렉터리 > 사용자 > 사용자 추가**로 이동하여 Tableau에서 만든 것과 동일한 사용자 이름(user@example.com)으로 사용자를 만듭니다.
2. 사용자가 만들어지면 해당 사용자에게 새 Okta 앱을 할당합니다. 사용자 이름을 클릭한 다음 **응용 프로그램 할당**에서 응용 프로그램을 할당합니다.

사전 인증을 위해 Mellon 설치

1. Apache 프록시 서버를 실행하는 EC2 인스턴스에서 다음 명령을 실행하여 PHP 및 Mellon 모듈을 설치합니다.

   ```
   sudo yum install httpd php mod_auth_mellon
   ```

2. `/etc/httpd/mellon` 디렉터리를 만듭니다.

Mellon을 사전 인증 모듈로 구성

두 프록시 서버에서 다음 절차를 실행합니다.

Okta 구성에서 만든 `pre-auth_idp_metadata.xml` 파일의 복사본이 있어야 합니다.

1. 디렉터리를 변경합니다.

   ```
   cd /etc/httpd/mellon
   ```

2. 서비스 공급자 메타데이터를 만듭니다. `mellon_create_metadata.sh` 스크립트를 실행합니다. 명령에 조직의 엔티티 ID와 반환 URL을 포함해야 합니다.

   반환 URL은 Okta의 *Single Sign On URL*이라고 합니다. 반환 URL의 최종 경로 요소는 이 절차의 뒷부분에서 설명하는 `mellon.conf` 구성 파일의

MellonEndpointPath를 일컫습니다. 이 예에서는 끝점 경로로 sso를 지정합니다.

예:

```
sudo /usr/libexec/mod_auth_mellon/mellon_create_metadata.sh
https://tableau.example.com "https://tableau.example.com/sso"
```

스크립트는 서비스 공급자 인증서, 키 및 메타데이터 파일을 반환합니다.

3. 쉽게 읽을 수 있도록 mellon 디렉터리에서 서비스 공급자 파일의 이름을 바꿉니다. 설명서에서 이러한 파일은 다음 이름으로 설명하겠습니다.

```
sudo mv *.key mellon.key
sudo mv *.cert mellon.cert
sudo mv *.xml sp_metadata.xml
```

4. pre-auth_idp_metadata.xml 파일을 동일한 디렉터리에 복사합니다.

5. /etc/httpd/conf.d 디렉터리에서 mellon.conf 파일을 만듭니다.

```
sudo nano /etc/httpd/conf.d/mellon.conf
```

6. 다음 내용을 mellon.conf에 복사합니다.

```
<Location />
MellonSPPrivateKeyFile /etc/httpd/mellon/mellon.key
MellonSPCertFile /etc/httpd/mellon/mellon.cert
MellonSPMetadataFile /etc/httpd/mellon/sp_metadata.xml
MellonIdPMetadataFile /etc/httpd/mellon/pre-auth_idp_
metadata.xml
MellonEndpointPath /sso
MellonEnable "info"
</Location>
```

7. 기존 tableau.conf 파일에 다음 콘텐츠를 추가합니다.

<VirtualHost *:80> 블록 안에 다음 콘텐츠를 추가합니다. 엔터티 ID의 공용 호스트 이름으로 ServerName 을 업데이트합니다.

```
DocumentRoot /var/www/html
ServerName tableau.example.com
ServerSignature Off
ErrorLog logs/error_sp.log
CustomLog logs/access_sp.log combined
LogLevel info
```

<VirtualHost *:80> 블록 밖에 Location 블록을 추가합니다. 최상위 도메인으로 MellonCookieDomain을 업데이트하여 표시된 쿠키 정보를 유지합니다.

```
<Location />
AuthType Mellon
MellonEnable auth
Require valid-user
MellonCookieDomain example.com
</Location>
```

전체 tableau.conf 파일은 다음 예제와 같아야 합니다.

```
<VirtualHost *:80>
ServerAdmin admin@example.com
ProxyHCExpr ok234 {%{REQUEST_STATUS} =~ /^[234]/}
Header add Set-Cookie "ROUTEID=.%{BALANCER_WORKER_ROUTE}e;
path=/" env=BALANCER_ROUTE_CHANGED
<Proxy balancer://tableau>
BalancerMember http://10.0.3.36/ route=1 hcmethod=GET
hcexpr=ok234 hcuri=/favicon.ico
BalancerMember http://10.0.4.15/ route=2 hcmethod=GET
hcexpr=ok234 hcuri=/favicon.ico
ProxySet stickysession=ROUTEID
</Proxy>
ProxyPreserveHost On
ProxyPass / balancer://tableau/
ProxyPassReverse / balancer://tableau/
```

```
DocumentRoot /var/www/html
ServerName tableau.example.com
ServerSignature Off
ErrorLog logs/error_sp.log
CustomLog logs/access_sp.log combined
LogLevel info
</VirtualHost>
<Location />
AuthType Mellon
MellonEnable auth
Require valid-user
MellonCookieDomain example.com
</Location>
```

8. 구성을 확인합니다. 다음 명령을 실행합니다.

   ```
   sudo apachectl configtest
   ```

   구성 테스트에서 오류가 반환되면 오류를 수정하고 configtest를 다시 실행합니다. 구성이 성공적으로 완료되면 `Syntax OK`가 반환됩니다.

9. httpd를 다시 시작합니다.

   ```
   sudo systemctl restart httpd
   ```

Okta에서 Tableau Server 응용 프로그램 만들기

1. Okta 대시보드에서: **응용 프로그램** > **응용 프로그램** > **앱 카탈로그 찾아보기**
2. **앱 통합 카탈로그 찾아보기**에서 `Tableau`를 검색하고 Tableau Server 타일을 선택한 다음 **추가**를 클릭합니다.
3. **Tableau Server 추가** > **일반 설정**에서 레이블을 입력하고 **다음**을 클릭합니다.
4. 로그온 옵션에서 **SAML 2.0**을 선택한 다음 고급 로그온 설정으로 스크롤합니다.
   - **SAML 엔터티 ID**: 공용 URL을 입력합니다(예: https://tableau.example.com).
   - **응용 프로그램 사용자 이름 형식**: 이메일

5. **ID 공급자 메타데이터** 링크를 클릭하여 브라우저를 시작합니다. 브라우저 링크를 복사합니다. 이 링크는 다음 절차에서 Tableau를 구성할 때 사용됩니다.
6. **완료**를 클릭합니다.
7. 사용자(user@example.com)에게 새 Tableau Server Okta 앱 할당: 사용자 이름을 클릭하고 **응용 프로그램 할당**에서 응용 프로그램을 할당합니다.

Tableau Server에서 IdP에 SAML을 사용하도록 설정

Tableau Server 노드 1에서 이 절차를 실행합니다.

1. Okta에서 Tableau Server 응용 프로그램 메타데이터를 다운로드합니다. 이전 절차에서 저장한 링크를 사용합니다.

   ```
   wget https://dev-66144217.okta.com/app/exk1egxgt1fhjkSeS5d7/sso/saml/metadata -O idp_metadata.xml
   ```

2. TLS 인증서와 관련 키 파일을 Tableau Server에 복사합니다. 키 파일은 RSA 키여야 합니다. SAML 인증서 및 IdP 요구 사항에 대한 자세한 내용은 *SAML 요구 사항* (Linux)을 참조하십시오.

   인증서 관리 및 배포를 간소화하고 보안을 위한 모범 사례를 적용하기 위해 신뢰할 수 있는 주요 외부 CA(인증 기관)에서 생성된 인증서를 사용할 것을 권장합니다. 또는 자체 서명 인증서를 생성하거나 TLS용 PKI의 인증서를 사용할 수도 있습니다.

   TLS 인증서가 없는 경우 아래 포함된 절차를 사용하여 자체 서명 인증서를 생성할 수 있습니다.

자체 서명 인증서를 생성합니다.

Tableau Server 노드 1에서 이 절차를 실행합니다.

a. 서명 루트 CA(인증 기관)키를 생성합니다.

```
openssl genrsa -out rootCAKey-saml.pem 2048
```

b. 루트 CA 인증서를 만듭니다.

```
openssl req -x509 -sha256 -new -nodes -key rootCAKey-
saml.pem -days 3650 -out rootCACert-saml.pem
```

인증서 필드에 값을 입력하라는 메시지가 표시됩니다. 예:

```
Country Name (2 letter code) [XX]:US
State or Province Name (full name) []:Washington
Locality Name (eg, city) [Default City]:Seattle
Organization Name (eg, company) [Default Company
Ltd]:Tableau
Organizational Unit Name (eg, section) []:Operations
Common Name (eg, your name or your server's hostname)
[]:tableau.example.com
Email Address []:example@tableau.com
```

c. 인증서 및 관련 키(아래 예에서 `server-saml.csr` 및 `server-saml.key`)를 만듭니다. 인증서의 주체 이름은 Tableau 호스트의 공용 호스트 이름과 일치해야 합니다. 주체 이름은 `-subj` 옵션과 `"/CN=<host-name>"` 형식을 사용하여 설정됩니다. 예:

```
openssl req -new -nodes -text -out server-saml.csr -keyout
server-saml.key -subj "/CN=tableau.example.com"
```

d. 위에서 만든 CA 인증서를 사용하여 새 인증서에 서명합니다. 다음 명령은 `crt` 형식으로도 인증서를 출력합니다.

```
openssl x509 -req -in server-saml.csr -days 3650 -CA
rootCACert-saml.pem -CAkey rootCAKey-saml.pem -
CAcreateserial -out server-saml.crt
```

e. 키 파일을 RSA로 변환합니다. Tableau에서 SAML을 사용하려면 RSA 키 파일이 필요합니다. 키를 변환하려면 다음 명령을 실행합니다.

```
openssl rsa -in server-saml.key -out server-saml-rsa.key
```

3. SAML을 구성합니다. 엔터티 ID와 반환 URL, 메타데이터 파일, 인증서 파일 및 키 파일에 대한 경로를 지정하여 다음 명령을 실행합니다.

```
tsm authentication saml configure --idp-entity-id
"https://tableau.example.com" --idp-return-url
"https://tableau.example.com" --idp-metadata idp_metadata.xml -
-cert-file "server-saml.crt" --key-file "server-saml-rsa.key"

tsm authentication saml enable
```

4. 조직에서 Tableau Desktop 2021.4 이상을 실행하는 경우 다음 명령을 실행하여 역방향 프록시 서버를 통한 인증을 사용하도록 설정해야 합니다.

   최상위 수준 도메인 쿠키 유지를 허용하도록 사전 인증 모듈(예: Mellon)을 구성한 경우 Tableau Desktop 버전 2021.2.1 ~ 2021.3은 이 명령을 실행하지 않고 작동합니다.

```
tsm configuration set -k features.ExternalBrowserOAuth -v false
```

5. 구성 변경 내용을 적용합니다.

```
tsm pending-changes apply
```

SAML 기능 확인

전체 SAML 기능을 확인하려면 이 절차를 시작할 때 만든 Tableau 관리자 계정으로 공용 URL(예: https://tableau.example.com)을 사용하여 Tableau Server에 로그인합니다.

확인 문제 해결

잘못된 요청: 이 시나리오의 일반적인 오류는 Okta의 "잘못된 요청" 오류입니다. 이 문제는 주로 브라우저가 이전 Okta 세션의 데이터를 캐싱할 때 발생합니다. 예를 들어 Okta 관리자 자격으로 Okta 응용 프로그램을 관리하고 다른 Okta 지원 계정을 사용하여 Tableau에 액세스하려고 하면 관리자 데이터의 세션 데이터가 "잘못된 요청" 오류를 발생시킬 수 있습니다. 로컬 브라우저 캐시를 지워도 이 오류가 계속되면 다른 브라우저로 연결하여 Tableau 시나리오의 유효성을 확인해 보십시오.

“Bad Request” 오류의 또 다른 원인은 Okta, Mellon 및 SAML 구성 프로세스 중에 입력한 많은 URL 중 하나의 입력 오류입니다. 이 모든 URL을 주의 깊게 확인하십시오.

오류를 야기한 URL은 주로 Apache 서버의 httpd `error.log` 파일에 명시됩니다.

찾을 수 없음 - 요청한 URL을 이 서버에서 찾지 못했습니다: 이 오류는 많은 구성 오류 중 하나를 나타냅니다.

사용자가 Okta로 인증된 후 이 오류가 발생하면 SAML을 구성할 때 Okta 사전 인증 응용 프로그램을 Tableau Server에 업로드한 것일 수 있습니다. Okta 사전 인증 응용 프로그램 메타데이터가 아니라 Tableau Server에 Okta Tableau Server 응용 프로그램 메타데이터가 구성되어 있는지 확인합니다.

다른 문제 해결 단계:

- 오타 또는 구성 오류가 있는지 `tableau.conf`를 주의 깊게 검토합니다.
- Okta 사전 인증 응용 프로그램 설정을 검토합니다. HTTP와 HTTPS 프로토콜이 이 항목에서 지정한 대로 설정되어 있는지 확인하십시오.
- 두 프록시 서버 모두에서 httpd를 다시 시작합니다.
- 두 프록시 서버 모두에서 `sudo apachectl configtest`가 "Syntax OK"를 반환하는지 확인합니다.
- 테스트 사용자가 Okta의 두 응용 프로그램에 모두 할당되었는지 확인합니다.
- 부하 분산 장치 및 연결된 대상 그룹에 연결 유지가 설정되어 있는지 확인합니다.

부하 분산 장치에서 Tableau Server로의 SSL/TLS 구성

일부 조직의 경우 클라이언트에서 백엔드 서비스에 이르는 엔드 투 엔드 암호화 채널을 요구합니다. 지금까지 설명한 기본 참조 아키텍처는 클라이언트에서 부하 분산 장치(조직의 웹 계층에서 실행됨)로의 SSL을 구성할 것을 명시합니다.

부하 분산 장치에서 Tableau Server로의 SSL을 구성하려면 다음을 수행해야 합니다.

- Tableau와 프록시 서버 모두에 유효한 SSL 인증서를 설치합니다.
- 부하 분산 장치에서 역방향 프록시 서버로의 SSL을 구성합니다.
- 프록시 서버에서 Tableau Server로의 SSL을 구성합니다.
- Tableau Server에서 PostgreSQL 인스턴스로의 SSL을 구성해도 됩니다.

이 항목의 나머지 부분에서는 예제 AWS 참조 아키텍처의 컨텍스트에서 이 구현을 설명합니다.

예제: AWS 참조 아키텍처에서 SSL/TLS 구성

이 섹션에서는 Tableau에서 SSL을 구성하고, 예제 AWS 참조 아키텍처에서 실행되는 모든 Apache 프록시 서버에서 SSL을 구성하는 방법을 설명합니다.

이 예 전반에 걸친 Linux 절차는 RHEL와 같은 배포에 대한 명령을 보여줍니다. 특히 여기에 나온 명령은 Amazon Linux 2 배포를 통해 개발되었습니다. Ubuntu 배포를 실행 중인 경우 그에 따라 명령을 편집합니다.

1단계: 인증서 및 관련 키 수집

인증서 관리 및 배포를 간소화하고 보안을 위한 모범 사례를 적용하기 위해 신뢰할 수 있는 주요 외부 CA(인증 기관)에서 생성된 인증서를 사용할 것을 권장합니다.

또는 자체 서명 인증서를 생성하거나 TLS용 PKI의 인증서를 사용할 수도 있습니다.

다음 절차는 자체 서명 인증서를 생성하는 방법을 설명합니다. 권장되는 타사 인증서를 사용하는 경우 이 절차를 건너뛰어도 됩니다.

프록시 호스트 중 하나에서 이 절차를 실행합니다. 인증서 및 연결된 키를 생성한 후 다른 프록시 호스트 및 Tableau Server 노드 1과 공유합니다.

1. 서명 루트 CA(인증 기관) 키를 생성합니다.

   ```
   openssl genrsa -out rootCAKey.pem 2048
   ```

2. 루트 CA 인증서를 만듭니다.

   ```
   openssl req -x509 -sha256 -new -nodes -key rootCAKey.pem -days
   3650 -out rootCACert.pem
   ```

   인증서 필드에 값을 입력하라는 메시지가 표시됩니다. 예:

   ```
   Country Name (2 letter code) [XX]:US
   State or Province Name (full name) []:Washington
   Locality Name (eg, city) [Default City]:Seattle
   Organization Name (eg, company) [Default Company Ltd]:Tableau
   Organizational Unit Name (eg, section) []:Operations
   Common Name (eg, your name or your server's hostname)
   []:tableau.example.com
   Email Address []:example@tableau.com
   ```

3. 인증서 및 관련 키(아래 예에서 `serverssl.csr` 및 `serverssl.key`)를 만듭니다. 인증서의 주체 이름은 Tableau 호스트의 공용 호스트 이름과 일치해야 합니다. 주체 이름은 `-subj` 옵션과 `"/CN=<host-name>"` 형식을 사용하여 설정됩니다. 예:

   ```
   openssl req -new -nodes -text -out serverssl.csr -keyout
   serverssl.key -subj "/CN=tableau.example.com"
   ```

4. 2단계에서 만든 CA 인증서를 사용하여 새 인증서에 서명합니다. 다음 명령은 `crt` 형식으로도 인증서를 출력합니다.

```
openssl x509 -req -in serverssl.csr -days 3650 -CA
rootCACert.pem -CAkey rootCAKey.pem -CAcreateserial -out
serverssl.crt
```

2단계: SSL에 대한 프록시 서버 구성

두 프록시 서버에서 다음 절차를 실행합니다.

1. Apache ssl 모듈을 설치합니다.

   ```
   sudo yum install mod_ssl
   ```

2. `/etc/ssl/private` 디렉터리를 만듭니다.

   ```
   sudo mkdir -p /etc/ssl/private
   ```

3. crt 및 key 파일을 다음 `/etc/ssl/` 경로에 복사합니다.

   ```
   sudo cp serverssl.crt /etc/ssl/certs/

   sudo cp serverssl.key /etc/ssl/private/
   ```

4. 다음 업데이트로 기존 `tableau.conf`를 업데이트합니다.

   - SSL 다시 쓰기 블록을 추가합니다.

     ```
     RewriteEngine on
     RewriteCond %{SERVER_NAME} =tableau.example.com
     RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI}
     [END,NE,R=permanent]
     ```

   - SSL 다시 쓰기 블록에서 `RewriteCond` 서버 이름을 업데이트합니다. 공용 호스트 이름(예: `tableau.example.com`)을 추가합니다.
   - `<VirtualHost *:80>`을 `<VirtualHost *:443>`으로 변경합니다.
   - `<VirtualHost *:443>` 및 `<Location />` 블록을 다음으로 래핑합니다. `<IfModule mod_ssl.c>` … `</IfModule>`.
   - `BalancerMember`: 프로토콜을 `http`에서 `https`로 변경합니다.

- SSL* 요소를 <VirtualHost *:443> 블록 안에 추가합니다.

```
SSLEngine on
SSLCertificateFile /etc/ssl/certs/serverssl.crt
SSLCertificateKeyFile /etc/ssl/private/serverssl.key
SSLProxyEngine on
SSLProxyVerify none
SSLProxyCheckPeerName off
SSLProxyCheckPeerExpire off
```

- LogLevel 요소에 ssl:warn을 추가합니다.
- 선택 사항: 인증 모듈을 설치하고 구성한 경우 tableau.conf 파일에 추가 요소가 포함될 수 있습니다. 예를 들어 <Location /> </Location> 블록에 요소가 포함됩니다.

여기에 SSL에 대해 구성된 예제 tableau.conf 파일이 나와 있습니다.

```
RewriteEngine on
RewriteCond %{SERVER_NAME} =tableau.example.com
RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI}
[END,NE,R=permanent]

<IfModule mod_ssl.c>
<VirtualHost *:443>
ServerAdmin admin@example.com
ProxyHCExpr ok234 {%{REQUEST_STATUS} =~ /^[234]/}
Header add Set-Cookie "ROUTEID=.%{BALANCER_WORKER_ROUTE}e;
path=/" env=BALANCER_ROUTE_CHANGED
<Proxy balancer://tableau>
BalancerMember https://10.0.3.36/ route=1 hcmethod=GET
hcexpr=ok234 hcuri=/favicon.ico
BalancerMember https://10.0.4.15/ route=2 hcmethod=GET
hcexpr=ok234 hcuri=/favicon.ico
ProxySet stickysession=ROUTEID
</Proxy>
ProxyPreserveHost On
ProxyPass / balancer://tableau/
```

```
ProxyPassReverse / balancer://tableau/
DocumentRoot /var/www/html
ServerName tableau.example.com
ServerSignature Off
ErrorLog logs/error_sp.log
CustomLog logs/access_sp.log combined
LogLevel info ssl:warn
SSLEngine on
SSLCertificateFile /etc/ssl/certs/serverssl.crt
SSLCertificateKeyFile /etc/ssl/private/serverssl.key
SSLProxyEngine on
SSLProxyVerify none
SSLProxyCheckPeerName off
SSLProxyCheckPeerExpire off
</VirtualHost>
<Location />
#If you have configured a pre-auth module (e.g. Mellon) include
those elements here.
</Location>
</IfModule>
```

5. 403 오류를 해결할 index.html 파일을 추가합니다.

```
sudo touch /var/www/html/index.html
```

6. httpd를 다시 시작합니다.

```
sudo systemctl restart httpd
```

3단계: 외부 SSL에 대한 Tableau Server 구성

프록시 1 호스트의 serverssl.crt 및 serverssl.key 파일을 초기 Tableau Server(노드 1)에 복사합니다.

노드 1에서 다음 명령을 실행합니다.

```
tsm security external-ssl enable --cert-file serverssl.crt --key-
file serverssl.key
tsm pending-changes apply
```

4단계: 선택적 인증 구성

Tableau에 대한 외부 ID 공급자를 구성한 경우 IdP 관리 대시보드의 반환 URL을 업데이트해야 할 수 있습니다.

예를 들어 Okta 사전 인증 응용 프로그램을 사용하는 경우 수신자 URL 및 대상 URL에 HTTPS 프로토콜을 사용하도록 응용 프로그램을 업데이트해야 합니다.

5단계: HTTPS에 대한 AWS 부하 분산 장치 구성

이 가이드에 설명된 대로 AWS 부하 분산 장치를 배포하는 경우 HTTPS 트래픽을 프록시 서버로 보내도록 AWS 부하 분산 장치를 다시 구성해야 합니다.

1. 기존 HTTP 대상 그룹을 등록 취소합니다.

   대상 그룹에서 부하 분산 장치에 구성된 HTTP 대상 그룹을 선택하고 **동작**을 클릭한 다음 **인스턴스 등록 및 등록 취소**를 클릭합니다.

   대상 등록 및 등록 취소 페이지에서 현재 구성된 인스턴스를 선택하고 **등록 취소**를 클릭한 다음 **저장**을 클릭합니다.

2. HTTPS 대상 그룹을 만듭니다.

   대상 그룹 > 대상 그룹 만들기

   - "인스턴스" 선택
   - 대상 그룹 이름 입력(예: `TG-internal-HTTPS`)
   - VPC 선택
   - 프로토콜: HTTPS 443
   - **상태 확인 > 고급 상태 확인 설정 > 성공 코드**에서 읽을 코드 목록(`200,303`)을 추가합니다.
   - **만들기**를 클릭합니다.

3. 방금 만든 대상 그룹을 선택한 다음 **대상** 탭을 클릭합니다.

- **편집**을 클릭합니다.
- 프록시 응용 프로그램을 실행 중인 EC2 인스턴스를 선택한 다음 **등록된 항목에 추가**를 클릭합니다.
- **저장**을 클릭합니다.

4. 대상 그룹이 만들어지면 연결 유지를 사용하도록 설정해야 합니다.

   - AWS 대상 그룹 페이지(**EC2 > 부하 분산 > 대상 그룹**)를 열고 방금 설정한 대상 그룹 인스턴스를 선택합니다. **동작** 메뉴에서 **특성 편집**을 선택합니다.
   - **특성 편집** 페이지에서 **연결 유지**를 선택하고 기간을 `1 day`로 지정한 다음 **변경 내용을 저장**합니다.

5. 부하 분산 장치에서 수신기 규칙을 업데이트합니다. 이 배포에 대해 구성한 부하 분산 장치를 선택한 다음 **수신기** 탭을 클릭합니다.

   - **HTTP:80**에서 **규칙 보기/편집**을 클릭합니다. 결과로 표시되는 **규칙** 페이지에서 편집 아이콘을 클릭하여 규칙을 편집합니다(페이지 맨 위에서 한 번 클릭하여 편집한 다음 규칙별로 다시 편집). 기존 THEN 규칙을 삭제하고 **동작 추가 > 리디렉션 대상...**을 클릭하여 바꿉니다. 결과로 나온 THEN 구성에서 `HTTPS` 및 포트 `443`을 지정하고 다른 옵션을 기본 설정으로 유지합니다. 설정을 저장한 다음 **업데이트**를 클릭합니다.
   - **HTTP:443**에서 **규칙 보기/편집**을 클릭합니다. 결과로 표시되는 **규칙** 페이지에서 편집 아이콘을 클릭하여 규칙을 편집합니다(페이지 맨 위에서 한 번 클릭하여 편집한 다음 규칙별로 다시 편집). **THEN** 구성의 **전달 대상...**에서 대상 그룹을 방금 만든 HTTPS 그룹으로 변경합니다. **그룹 수준 연결 유지**에서 연결 유지를 사용하도록 설정하고 기간을 1일로 설정합니다. 설정을 저장한 다음 **업데이트**를 클릭합니다.

6단계: SSL 확인

https://tableau.example.com으로 이동하여 구성을 확인합니다.