aws

자동화된 설치 가이드

Wickr Enterprise

Wickr Enterprise: 자동화된 설치 가이드

Table of Contents

Wickr Enterprise는 무엇입니까?

Wickr Enterprise는 조직과 정부 기관이 일대일 및 그룹 메시지, 음성 및 영상 통화, 파일 공유, 화면 공유 등을 통해 안전하게 통신할 수 있도록 지원하는 종단 간 암호화 서비스입니다. 고객은 Wickr Enterprise를 사용하여 소비자용 메시징 앱과 관련된 데이터 보존 의무를 극복하고 안전하게 협업을 촉진할 수 있습니다. 고급 보안 및 관리 제어를 통해 조직은 법률 및 규제 요구 사항을 충족하고 데이터 보안 문제에 대한 맞춤형 솔루션을 구축할 수 있습니다.

보존 및 감사 목적으로 고객이 제어하는 개인 데이터 스토어에 정보를 기록할 수 있습니다. 고객은 권한 설정, 임시 메시징 옵션 구성, 보안 그룹 정의 등 데이터에 대한 포괄적인 관리 제어를 할 수 있습니다. 또한 관리자는 Wickr 봇을 사용하여 워크플로를 안전하게 자동화할 수 있습니다. Wickr Enterprise는 OpenID Connect(OIDC)를 통해 활성화된 디렉터리 및 Single Sign-On(SSO)과 같은 추가 서비스와 통합됩니다. Wickr Enterprise 구성을 시작하려면 Wickr Enterprise 시작하기를 참조하십시오.

> **Note**
>
> Wickr Enterprise 배포 패키지가 아직 없는 경우 비즈니스 문의를 참조하십시오.

Wickr Enterprise 사용 시작하기

주제

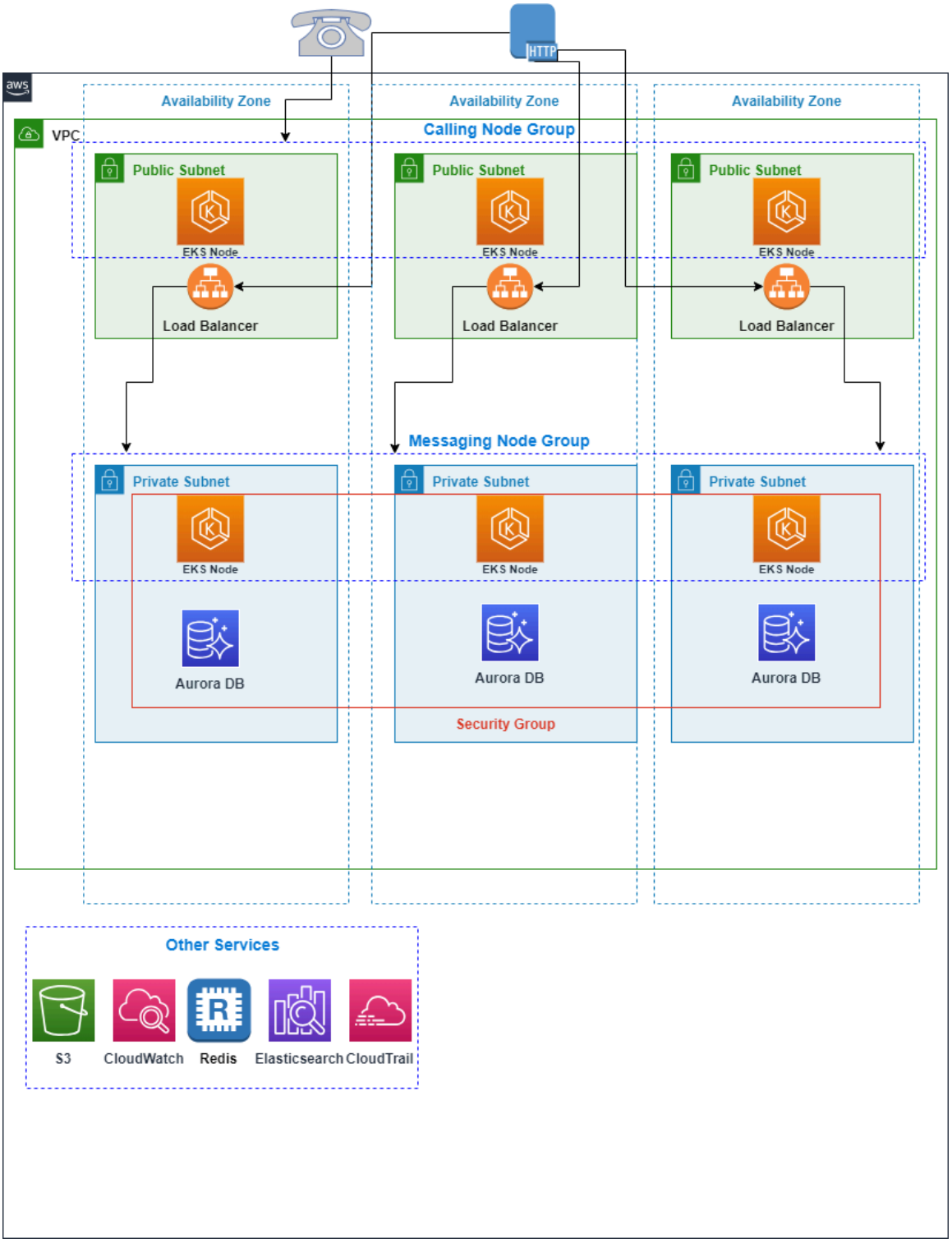
요구 사항

시작하기 전, 다음 요구 사항을 충족하는지 확인하십시오.

- Node.js 16+ 다운로드
- AWS CLI 계정의 자격 증명으로 구성됩니다.

  이는 `~/.aws/config`의 구성 파일에서 가져오거나 `AWS_` 환경 변수를 사용하여 제공됩니다.
- kubectl 설치 자세한 내용은 Amazon EKS 사용 설명서의 kubectl 설치 및 업데이트를 참조하십시오.
- kots CLI를 설치합니다. 자세한 내용은 kots CLI 설치 단원을 참조하십시오.
- 허용 목록에 대한 포트: HTTPS 및 TCP 호출 트래픽의 경우 443/TCP, UDP 호출 트래픽의 경우 16384-19999/UDP, TCP/8443

아키텍처



종속성을 설치합니다.

다음의 명령을 사용하여 모든 종속성을 기본 패키지에 추가할 수 있습니다.

```
npm install
```

구성

AWS Cloud Development Kit (AWS CDK) 는 컨텍스트 값을 사용하여 애플리케이션의 구성을 제어합니다. Wickr Enterprise는 CDK 컨텍스트 값을 사용하여 Wickr Enterprise 설치의 도메인 이름 또는 RDS 백업 보존 일수와 같은 설정을 제어할 수 있도록 합니다. 자세한 내용을 알아보려면 AWS Cloud Development Kit (AWS CDK) 개발자 안내서의 런타임 컨텍스트를 참조하십시오.

컨텍스트 값을 설정하는 방법은 여러 가지가 있지만 특정 사용 사례에 맞게 `cdk.context.json`의 값을 편집하는 것이 좋습니다. `wickr/` 로 시작하는 컨텍스트 값만 Wickr Enterprise 배포와 관련이 있고 나머지는 CDK 관련 컨텍스트 값입니다. 다음 번에 CDK를 통해 업데이트할 때 동일한 설정을 유지하려면 이 파일을 저장하십시오.

최소한 , `wickr/domainName`및 `wickr/acm:certificateArn` 또는 `wickr/route53:hostedZoneId` 및 `wickr/licensePath`를 설정해야 합니다`wickr/route53:hostedZoneName`.

퍼블릭 호스팅 영역 사용

에 Route 53 퍼블릭 호스팅 영역이 있는 경우 다음 설정을 사용하여 CDK 컨텍스트를 구성하는 AWS 계정것이 좋습니다.

- `wickr/domainName`- 이 Wickr Enterprise 배포에 사용할 도메인 이름. Route 53 퍼블릭 호스팅 영역을 사용하는 경우, 이 도메인 이름에 대한 DNS 레코드 및 ACM 인증서가 자동으로 생성됩니다.
- `wickr/route53:hostedZoneName`- DNS 레코드를 생성할 Route 53 호스팅 영역 이름
- `wickr/route53:hostedZoneId`- DNS 레코드를 생성할 Route 53 호스팅 영역 ID

이 방법은 Wickr Enterprise 배포 앞에 있는 로드 밸런서에 대한 도메인 이름을 가리키는 DNS 레코드와 함께 사용자를 대신하여 ACM 인증서를 생성합니다.

퍼블릭 호스팅 영역 없음

계정에 Route 53 퍼블릭 호스팅 영역이 없는 경우 ACM 인증서를 수동으로 생성하고 `wickr/acm:certificateArn` 컨텍스트 값을 사용하여 CDK로 가져와야 합니다.

- `wickr/domainName`- 이 Wickr Enterprise 배포에 사용할 도메인 이름. Route 53 퍼블릭 호스팅 영역을 사용하는 경우, 이 도메인 이름에 대한 DNS 레코드 및 ACM 인증서가 자동으로 생성됩니다.

- `wickr/acm:certificateArn`- 로드 밸런서에서 사용할 ACM 인증서의 ARN. 계정에서 Route 53 퍼블릭 호스팅 영역을 사용할 수 없는 경우 이 값을 제공해야 합니다.

ACM으로 인증서 가져오기

다음 명령을 사용하여 외부에서 가져온 인증서를 가져올 수 있습니다.

```
aws acm import-certificate \
   --certificate fileb://path/to/cert.pem \
   --private-key fileb://path/to/key.pem \
   --certificate-chain fileb://path/to/chain.pem
```

출력은 `wickr/acm:certificateArn` 컨텍스트 설정 값으로 사용해야 하는 인증서 ARN이 됩니다. 업로드한 인증서가 `wickr/domainName` 에 유효한지 확인해야 합니다. 그렇지 않으면 HTTPS 연결에서 유효성을 검사할 수 없습니다. 자세한 내용은 AWS Certificate Manager 사용 설명서에서 인증서 가져오기를 참조하십시오.

DNS 레코드 생성

사용할 수 있는 퍼블릭 호스팅 영역이 없으므로 배포가 완료된 후 Wickr Enterprise 배포 앞에 있는 로드 밸런서를 가리키도록 DNS 레코드를 수동으로 생성해야 합니다.

기존 VPC에 배포

기존 VPC를 사용해야 하는 경우 사용할 수 있습니다. 그러나 VPC는 EKS에 필요한 사양을 충족하도록 구성해야 합니다. 자세한 내용은 Amazon EKS 사용 설명서의 VPC 및 서브넷에 대한 Amazon EKS 네트워킹 요구 사항 보기를 참조하고 사용할 VPC가 이러한 요구 사항을 충족하는지 확인하세요.

또한 다음 서비스에 대한 VPC 엔드포인트가 있는지 확인하는 것이 좋습니다.

- CLOUDWATCH
- CLOUDWATCH_LOGS
- EC2
- EC2_MESSAGES
- ECR
- ECR_DOCKER
- ELASTIC_LOAD_BALANCING

- KMS
- SECRETS_MANAGER
- SSM
- SSM_MESSAGES

기존 VPC에 리소스를 배포하려면 다음 컨텍스트 값을 설정합니다.

- `wickr/vpc:id` - 리소스를 배포할 VPC의 ID입니다(예 `vpc-412beef`).
- `wickr/vpc:cidr` - VPC의 IPv4 CIDR(예 `172.16.0.0/16`).
- `wickr/vpc:publicSubnetIds` - VPC에 있는 퍼블릭 서브넷의 쉼표로 구분된 목록입니다. Application Load Balancer 및 호출 EKS 작업자 노드는 이러한 서브넷 (예:`subnet-6ce9941,subnet-1785141,subnet-2e7dc10`)에 배포됩니다.
- `wickr/vpc:privateSubnetIds` - VPC에 있는 퍼블릭 서브넷의 쉼표로 구분된 목록입니다. EKS 작업자 노드와 bastion 서버는 이러한 서브넷 (예: `subnet-f448ea8,subnet-3eb0da4,subnet-ad800b5`)에 배포됩니다.
- `wickr/vpc:isolatedSubnetIds` - VPC에 있는 격리된 서브넷의 쉼표로 구분된 목록입니다. RDS 데이터베이스는 이러한 서브넷 (예: `subnet-d1273a2,subnet-33504ae,subnet-0bc83ac`)에 배포됩니다.
- `wickr/vpc:availabilityZones`- VPC에 있는 서브넷의 쉼표로 구분된 가용 영역 목록 (예: `us-east-1a,us-east-1b,us-east-1c`).

인터페이스 VPC 엔드포인트에 대한 자세한 내용은 인터페이스 VPC 엔드포인트를 사용하여 AWS 서비스 액세스를 참조하세요.

기타 설정

자세한 내용은 컨텍스트 값을 참조하십시오.

부트스트랩

이 특정 AWS 계정 및 리전에서 CDK를 처음 사용하는 경우 먼저 계정을 부트스트랩하여 CDK 사용을 시작해야 합니다.

```
npx cdk bootstrap
```

배포

이 프로세스는 약 45분이 소요됩니다.

```
npx cdk deploy --all --require-approval=never
```

설치가 완료되면 인프라가 생성되고 Wickr Enterprise 설치를 시작할 수 있습니다.

DNS 레코드 생성

CDK를 구성할 때 퍼블릭 호스팅 영역을 사용한 경우에는 이 단계가 필요하지 않습니다.

배포 프로세스의 출력에는 로드 밸런서의 DNS 이름인 값`WickrAlb.AlbDnsName`이 포함됩니다. 출력값은 다음과 같습니다.

```
WickrAlb.AlbDnsName = Wickr-Alb-1Q5IBPJR4ZVZR-409483305.us-west-2.elb.amazonaws.com
```

이 경우 DNS 이름은 `Wickr-Alb-1Q5IBPJR4ZVZR-409483305.us-west-2.elb.amazonaws.com`입니다. 이 값은 도메인 이름에 대한 CNAME 또는 A/AAAA(ALIAS) 레코드를 만들 때 사용해야 하는 값입니다.

배포 출력이 없는 경우 다음 명령을 실행하여 로드 밸런서 DNS 이름을 표시합니다.

```
aws cloudformation describe-stacks --stack-name WickrAlb \
   --query 'Stacks[0].Outputs[?OutputKey==`AlbDnsName`].OutputValue' \
   --output text
```

KOTS Config 생성

> **Warning**
>
> 이 파일에는 설치에 대한 민감한 정보가 들어 있습니다. 공개적으로 공유하거나 저장하지 마십시오.

Wickr Enterprise 설치 프로그램을 성공적으로 설치하려면 인프라에 대한 여러 구성 값이 필요합니다. 도우미 스크립트를 사용하여 구성 값을 생성할 수 있습니다.

```
./bin/generate-kots-config.ts > wickr-config.json
```

첫 단계에서 외부 인증서를 ACM으로 가져온 경우 `--ca-file` 플래그를 이 스크립트에 전달하십시오. 예를 들면 다음과 같습니다.

```
./bin/generate-kots-config.ts --ca-file path/to/chain.pem > wickr-config.json
```

스택이 존재하지 않는다는 오류 메시지가 표시되면 `AWS_REGION`환경 변수(`export AWS_REGION=us-west-2`)를 선택한 지역으로 설정하고 다시 시도하십시오. 또는 컨텍스트 값를 설정하는 경우 `--stack-suffix` 플래그와 함께 접미사를 `wickr/stackSuffix`전달합니다.

Kubernetes 클러스터에 연결

Amazon EKS API는 배포의 일부로 생성된 Bastion Host를 통해서만 액세스할 수 있습니다. 그 결과, 모든 kubectl 명령은 Bastion Host 자체에서 실행되거나 Bastion Host를 통해 프록시되어야 합니다.

Bastion을 통한 프록시 연결

클러스터에 처음 연결할 때는 aws eks update-kubeconfig 명령을 사용하여 로컬 kubeconfig 파일을 업데이트한 다음 구성에서 proxy-url을 설정해야 합니다. 그런 다음 클러스터에 연결할 때마다 Bastion Host와 SSM 세션을 시작하여 API 액세스를 위해 프록시로 포트 포워딩합니다.

일회성 설정

WickrEksCloudFormation 스택에는 WickrEnterpriseConfigCommand로 시작하는 이름을 가진 출력 값이 있습니다. 이 값에는 클러스터의 kubectl 구성을 생성하는 데 필요한 전체 명령이 포함됩니다. 이 출력은 다음 명령으로 확인할 수 있습니다.

```
aws cloudformation describe-stacks --stack-name WickrEks \
--query 'Stacks[0].Outputs[?starts_with(OutputKey,
 `WickrEnterpriseConfigCommand`)].OutputValue' \
--output text
```

그러면 aws eks update-kubeconfig로 시작하는 명령이 출력되어야 합니다. 이 명령을 실행합니다.

다음으로, Bastion Host를 통한 프록시 요청으로 Kubernetes 구성을 수정해야 합니다. 이 작업은 다음 명령을 사용하여 완료될 수 있습니다.

```
CLUSTER_ARN=$(aws cloudformation describe-stacks --stack-name WickrEks --query
 'Stacks[0].Outputs[?OutputKey==`WickrEnterpriseEksClusterArn`].OutputValue' --output
 text)
kubectl config set "clusters.${CLUSTER_ARN}.proxy-url" http://localhost:8888
```

제대로 작업했으면, 'Property "clusters.arn:aws:eks:us-west-2:012345678912:cluster/WickrEnterprise5B8BF472-1234a41c4ec48b7b615c6789d93dcce.proxy-url" set.'과 같은 출력이 표시됩니다.

Bastion으로 향하는 포트 포워드

Amazon EKS 클러스터에 연결하려면 Bastion Host에서 실행되는 프록시로 요청을 포워딩하는 SSM 세션을 시작해야 합니다. 이 작업을 수행하기 위한 명령은 `WickrEks` 스택의 출력으로 제공됩니다`BastionSSMProxyEKSCommand`. 다음 명령을 실행하여 출력 값을 확인합니다.

```
aws cloudformation describe-stacks --stack-name WickrEks \
--query 'Stacks[0].Outputs[?OutputKey==`BastionSSMProxyEKSCommand`].OutputValue' \
--output text
```

출력되는 명령은 `aws ssm start-session`로 시작됩니다. 이 명령을 실행하여 포트 8888에서 실행되는 로컬 프록시를 시작합니다. 이 포트를 통해 Amazon EKS 클러스터에 연결할 수 있습니다. 포트 전달이 제대로 작동하면 출력에 '연결 대기 중... '이 표시되어야 합니다. Amazon EKS 클러스터에 액세스하는 데 필요한 시간 내내 이 프로세스를 계속 실행되도록 하십시오.

모든 것이 올바르게 설정된 경우 다른 터미널`kubectl get nodes`에서를 실행하여 Amazon EKS 클러스터의 작업자 노드를 나열할 수 있습니다.

```
kubectl get nodes
   NAME                             STATUS   ROLES    AGE     VERSION
   ip-10-0-111-216.ec2.internal   Ready     none   3d      v1.26.4-eks-0a21954
   ip-10-0-180-1.ec2.internal     Ready     none   2d23h   v1.26.4-eks-0a21954
   ip-10-0-200-102.ec2.internal   Ready     none   3d      v1.26.4-eks-0a21954
```

Wickr Enterprise 설치

사용자가 Kubernetes 클러스터에 연결된 후에, `kubectl kots` 플러그인을 사용하여 Wickr Enterprise 설치를 시작할 수 있습니다. KOTS 라이선스 파일(Wickr에서 제공하는 `.yaml`파일)과 Config Values 값 파일이 필요할 것이며, 이 파일은 KOTS 구성 생성 섹션의 파일 `wickr-config.json`에 저장되어 있습니다. KOTS 구성 생성에 대한 자세한 내용은 KOTS Config 생성을 참조하십시오.

Wickr Enterprise 수동 설치

다음 명령을 실행하면 Wickr 엔터프라이즈 설치가 시작됩니다.

```
kubectl kots install wickr-enterprise-ha \
    --license-file ./license.yaml \
    --config-values ./wickr-config.json  \
    --namespace wickr \
    --skip-preflights
```

KOTS 관리 콘솔의 비밀번호를 입력하라는 메시지가 표시됩니다. 나중에 Wickr Enterprise 설치 구성을 업그레이드하거나 변경할 때 필요하므로 이 비밀번호를 저장해 두십시오.

설치가 완료되면, `kubectl kots`은 KOTS 관리 콘솔에 액세스할 수 있는 로컬 포트(보통`http://localhost:8080`)가 열 것입니다. 이 사이트에서 Wickr Enterprise 설치 상태를 변경 또는 모니터링하거나 브라우저에서 설치하도록 구성한 도메인 이름을 방문하여 Wickr 설정을 시작할 수 있습니다.

Lambda를 사용하여 Wickr Enterprise 설치

CDK 배포 중에 Lambda가 생성되고 호출되어 사용자를 대신하여 Wickr Enterprise 설치를 자동으로 완료합니다. 수동으로 호출하려면 AWS 콘솔을 열고 `WickrLambda-func*` Lambda 함수를 찾고 테스트 탭에서를 선택합니다. `test`입력은 관련이 없습니다.

사후 설치

Wickr Enterprise 설치를 관리하는 데 사용할 수 있는 두 개의 웹 콘솔이 있으며 하나는 KOTS 관리 콘솔과 다른 하나는 Wickr 관리 콘솔입니다.

> **Note**
>
> 조직의 백업 및 로깅 정책(Amazon S3 설정, Elastic Load Balancing 액세스 로그, Amazon Virtual Private Cloud 흐름 로그)을 반영하는 데 필요한 사항을 변경하십시오.

KOTS 관리 콘솔

이 인터페이스는 Wickr Enterprise의 배포된 버전을 관리하는 데 사용됩니다. 설치 상태를 확인하고, 구성을 수정하거나, 업그레이드를 수행할 수 있습니다. KOTS 관리 콘솔은 다음 명령을 사용해야만 열 수 있는 Kubernetes 포트 포워드를 통해서만 액세스할 수 있습니다.

```
kubectl kots --namespace wickr admin-console
```

> **Note**
>
> 먼저 Bastion으로 포트 포워딩 섹션에서 설명된 대로 Bastion 연결을 설정해야 합니다. Bastion으로의 포트 포워딩에 대한 자세한 내용은 을 Bastion을 통한 프록시 연결을 참조하십시오.

포트 포워드가 성공적으로 구성되면 이전 명령은 다음을 출력합니다.

```
• Press Ctrl+C to exit
• Go to http://localhost:8800 to access the Admin Console
```

제공된 URL을 사용하여 KOTS 관리 콘솔에 액세스합니다. 로그인 비밀번호는 설치 중에 `kubectl kots install`을 실행할 때 선택한 비밀번호입니다. 비밀번호를 재설정해야 하는 경우 KOTS 관리 콘솔 비밀번호 재설정을 참조하십시오.

Wickr 관리 콘솔

이 인터페이스는 Wickr Enterprise 설치를 구성하여 네트워크, 사용자 및 페더레이션을 설정하는 데 사용됩니다. 로드 밸런서를 가리키도록 구성한 DNS 이름으로 HTTPS를 통해 액세스할 수 있습니다. 공용 호스팅 영역을 사용하여 DNS를 자동으로 구성한 경우 도메인 이름은 `wickr/domainName` 컨텍스트 값의 값입니다.

기본 사용자 이름은 `admin`이고 비밀번호는 `Password123`입니다. 처음 로그인할 때 이 비밀번호를 변경해야 합니다.

컨텍스트 값

컨텍스트 값은 앱, 스택 또는 구성과 연결할 수 있는 키-값 페어입니다. 파일(보통 프로젝트 디렉터리 내 `cdk.json` 또는 `cdk.context.json`) 또는 명령줄로부터 사용자의 앱에 제공될 수 있습니다. CDK는 컨텍스트 값을 사용하여 애플리케이션 구성을 제어합니다. Wickr Enterprise는 CDK 컨텍스트 값을 사용하여 Wickr Enterprise 설치의 도메인 이름 또는 RDS 백업 보존 일수와 같은 설정을 제어할 수 있도록 합니다.

컨텍스트 값을 설정하는 방법은 여러 가지가 있지만 특정 사용 사례에 맞게 `cdk.context.json`의 값을 편집하는 것이 좋습니다. `wickr/`로 시작하는 컨텍스트 값만이 Wickr Enterprise 배포와 관련이 있습니다.

이름	설명	기본값
`wickr/licensePath`	KOTS 라이선스 경로 (Wickr에서 제공하는 `.yaml` 파일).	null
`wickr/domainName`	이 Wickr Enterprise 배포에 사용할 도메인 이름입니다. Route 53 퍼블릭 호스팅 영역을 사용하는 경우, 이 도메인 이름에 대한 DNS 레코드 및 ACM 인증서가 자동으로 생성됩니다.	null
`wickr/route53:hostedZoneId`	DNS 레코드를 생성할 Route 53 호스팅 영역 ID입니다.	null
`wickr/route53:hostedZoneName`	DNS 레코드를 생성할 Route 53 호스팅 영역 이름	null
`wickr/acm:certificateArn`	로드 밸런서에서 사용할 ACM 인증서의 ARN 계정에서 Route 53 퍼블릭 호스팅 영역을 사용할 수 없는 경우 이 값을 제공해야 합니다.	null

이름	설명	기본값
wickr/caPath	인증서 경로, 자체 서명된 인증서를 사용할 때만 필요합니다.	null
wickr/vpc:id	리소스를 배포할 VPC의 ID입니다. 기존 VPC에 배포할 때만 필요합니다. 설정하지 않으면 새 VPC가 생성됩니다.	null
wickr/vpc:cidr	생성된 VPC와 연결할 IPv4 CIDR 블록입니다. 기존 VPC에 배포하는 경우 이를 기존 VPC의 CIDR로 설정하십시오.	172.16.0.0/16
wickr/vpc:availabilityZones	쉼표로 구분된 가용 영역 목록입니다. 기존 VPC에 배포할 때만 필요합니다.	null
wickr/vpc:publicSubnetIds	쉼표로 구분된 퍼블릭 서브넷 ID의 목록입니다. 기존 VPC에 배포할 때만 필요합니다.	null
wickr/vpc:privateSubnetIds	쉼표로 구분된 프라이빗 서브넷 ID의 목록입니다. 기존 VPC에 배포할 때만 필요합니다.	null
wickr/vpc:isolatedSubnetIds	RDS 데이터베이스의 격리된 서브넷 ID의 목록입니다. 기존 VPC에 배포할 때만 필요합니다.	null
wickr/rds:deletionProtection	RDS 인스턴스에서 삭제 보호를 활성화합니다.	true
wickr/rds:removalPolicy	RDS 인스턴스 '스냅샷', '제거' 또는 '보존'에 대한 제거 정책	snapshot

이름	설명	기본값
wickr/rds:readerCount	RDS 클러스터에서 생성할 리더 인스턴스의 수	1
wickr/rds:instanceType	RDS 인스턴스에 사용할 인스턴스 유형.	r6g.xlarge
wickr/rds:backupRetentionDays	백업을 보존할 일수입니다.	7
wickr/eks:namespace	EKS의 Wickr 서비스에 대한 기본 네임스페이스입니다.	Wickr
wickr/eks:defaultCapacity	메시징 인프라용 EKS 워커 노드 수	3
wickr/eks:defaultCapacityCalling	통화 인프라용 EKS 워커 노드 수	2
wickr/eks:instanceTypes	Messaging EKS 워커 노드에 사용할 인스턴스 유형의 쉼표로 구분된 목록입니다.	m5.xlarge
wickr/eks:instanceTypesCalling	EKS 워커 노드 호출에 사용할 인스턴스 유형의 쉼표로 구분된 목록입니다.	c5n.large
wickr/eks:enableAutoscaler	EKS의 클러스터 오토스케일러 기능을 활성화하도록 전환합니다.	true
wickr/s3:expireAfterDays	파일 업로드가 S3 버킷에서 제거되기까지 경과한 일수입니다.	1095

이름	설명	기본값
`wickr/eks:clusterVersion`	Kubernetes 버전, kubectlLayer 버전, albController 버전, nodeGroupRelease 버전 등을 포함한 클러스터 버전.	1.27
`wickr/stackSuffix`	CloudFormation 스택 이름에 적용할 접미사입니다.	"
`wickr/autoDeployWickr`	Lambda를 사용하여 Wickr 애플리케이션을 자동 배포합니다.	true

리소스 폐기

이 AWS CDK 애플리케이션에서 생성한 모든 항목을 삭제하려면 다른 모든 WickrRds 스택보다 먼저 스택을 삭제해야 합니다.

Amazon RDS 리소스를 제대로 삭제하려면, 삭제 보호를 비활성화하고 제거 정책을 snapshot 또는 destroy 중 하나로 설정해야 합니다. 이들이 현재 설정이 아닌 경우, wickr/rds:deletionProtection 및 wickr/rds:removalPolicy 값을 AWS CDK 컨텍스트 내에서 수정하고, npx cdk deploy -e WickrRds를 실행하여 Amazon RDS 스택을 재배포하십시오.

일단 삭제 보호 및 제거 정책이 적절하게 설정되면 WickrRds 스택에 대해 cdk destroy를 실행합니다.

```
npx cdk destroy WickrRds
```

WickrRds 스택이 폐기를 완료하면 다음 명령을 사용하여 나머지 CloudFormation 스택을 제거할 수 있습니다.

```
npx cdk destroy --all
```

문제 해결

Wickr 네임스페이스 삭제

다시 시작하기 위해 `wickr` 네임스페이스를 삭제해야 하는 경우 먼저 해당 네임스페이스 내에서 CDK로 만든 모든 서비스 계정을 백업해야 합니다. 이러한 서비스 계정을 사용하면 Wickr 서비스가 IAM 역할을 통해 AWS APIs. 이러한 기능이 없으면 Amazon Simple Storage Service(S3)를 통한 파일 업로드와 같은 작업은 더 이상 작동하지 않습니다.

다음 명령을 사용하여 서비스 계정을 백업하고 `wickr` 네임스페이스와 적절한 서비스 계정을 삭제 및 다시 생성합니다.

```
kubectl -n wickr get sa fileproxy -o yaml > fileproxy-sa.yaml && \
  kubectl delete ns wickr && \
  kubectl create ns wickr && \
  kubectl apply -f fileproxy-sa.yaml
```

KOTS 관리 콘솔 비밀번호 재설정

다음 명령어를 사용하여 KOTS 관리 콘솔 비밀번호를 재설정할 수 있습니다.

```
kubectl kots -n wickr reset-password
```

이 암호를 변경하면 일반적으로 자동화에서 다시 사용되지 않지만 `wickr/kots` Secrets Manager 보안 암호도 업데이트할 수 있습니다.

Bastion을 사용하여 EKS 클러스터에 연결하는 문제

Bastion을 통해 EKS 클러스터에 연결하는 속도가 느리거나 가끔 시간 초과되는 경우 `kubectl` 명령을 실행할 때 다음 오류가 표시될 수 있습니다.

net/http: 연결을 기다리는 동안 요청이 취소됨(헤더를 기다리는 동안Client.Timeout 초과)

이 문제는 종종 SSM을 통해 접속 호스트`BastionSSMCommand`에 로그인하고(WickrEks 스택의 참조) `tinyproxy` 서비스를 다시 시작하여 해결할 수 있습니다.

```
sudo systemctl restart tinyproxy
```

사용자 지정 설치

사용자 지정 설치 섹션에서 Wickr Enterprise를 설치하는 방법을 알아봅니다.

주제

요구 사항

Wickr Enterprise 설치를 시작하기 전에 다음 요구 사항이 충족되는지 확인합니다.

하드웨어 요구 사항

Wickr Enterprise를 사용하려면 Kubernetes 클러스터가 필요합니다. 리소스 부족 모드가 활성화된 단일 노드에서 작동할 수 있지만 일반적인 프로덕션 용도로는 권장되지 않습니다. 프로덕션 배포에서는 최소 3개의 메시징 작업자 노드와 최소 2개의 호출 작업자 노드를 사용하는 것이 좋습니다.

작업자 노드에는 다음과 같은 최소 사양이 있어야 합니다.

- CPU 코어 2~4개
- 8GB 램
- 200GB의 디스크 공간

최소 하드웨어 요구 사항

낮은 리소스 모드에서 실행되는 단일 작업자 노드 클러스터에는 최소 3,000m CPU와 5846Mi Ram이 필요합니다. 여기에는 kube 시스템 포드가 포함되지 않습니다.

포드별 리소스 요구 사항

포드 이름	소유자	CPU	Memory
admin-api	Wickr	1억	256Mi
directory	Wickr	1억	128Mi
만료자	Wickr	1억	128Mi
fileproxy	Wickr	1억	256Mi
oidc	Wickr	1억	128Mi
opensearch	Wickr	5억	100Mi
orville	Wickr	50분	128Mi
orville-redis	Wickr	50분	128Mi
push-device	Wickr	1억	128Mi
rabbitmq	Wickr	50분	256Mi
반응	Wickr	1억	64Mi
영수증	Wickr	250분	128Mi
redis	Wickr	50분	128Mi
server-api	Wickr	250분	256Mi

포드 이름	소유자	CPU	Memory
스위치보드	Wickr	250분	512Mi
코차드름	KOTS	50분	50Mi
코츠담-미니오	KOTS	1억	512Mi
kotsadm-rqlite	KOTS	200m	1Gi
미니 운영자	내부 S3	200m	256Mi
미니 테넌트	내부 S3	1억	256Mi
mysql-기본	내부 MySQL	1억	512Mi
mysql-보조	내부 MySQL	1억	512Mi

스토리지 요구 사항

Wickr Enterprise는 영구 볼륨 클레임을 생성할 때 사용할 기본 StorageClass가 필요합니다. 에어 갭 환경 또는 온프레미스에서 배포하는 경우 클러스터에 대해 하나를 구성해야 할 수 있습니다. 사용 가능한 옵션 중 하나는 Longhorn입니다. 권장 디스크 공간 요구 사항은 내부 S3 옵션 및 내부 Mysql 옵션의 사용과 파일 업로드에 사용할 수 있는 공간에 따라 달라집니다.

- 내부 이미지 캐싱: ~60Gi
- RabbitMQ: 리소스 부족 모드에서 기본값 24Gi/8Gi
- Redis: 리소스 부족 모드에서 기본값 24Gi/8Gi
- OpenSearch: 리소스 부족 모드에서 기본값 24Gi/8Gi
- 내부 Mysql: 리소스 부족 모드에서 기본값 80Gi/20Gi
- 내부 S3: 리소스 부족 모드에서 160Gi 기본값/2Gi
- KOTS 미니오: 4Gi
- KOTS Rqlite: 1Gi

최소 스토리지 크기

- 내부 S3 및 내부 Mysql을 사용하는 377Gi 기본값

- 리소스 부족 모드에서 111Gi

Kubernetes 버전 요구 사항

Wickr Enterprise는 복제된 KOTS를 사용합니다. 상용 소프트웨어 배포 플랫폼인 Replicated는 현재 지원되는 Kubernetes 버전 목록을 제공합니다. 자세한 내용은 Kubernetes 버전 호환성을 참조하세요.

소프트웨어 요구 사항

Wickr Enterprise를 사용하려면 Kubernetes 클러스터와 KOTS가 필요합니다. 지원되는 OS 및 Kubernetes 버전은 KOTS 설명서를 참조하세요. 자세한 내용은 최소 시스템 요구 사항을 참조하세요.

개발자 호스트 시스템

운영 체제 -이 설명서의 명령은 WSL(Linux용 Windows 하위 시스템)이 설치된 Linux, MacOS 또는 Windows에서 작동하도록 설계되었습니다.

내부 상태 저장 서비스

Wickr Enterprise는 MySQL 데이터베이스와 S3 호환 스토리지 모두에 내부 서비스를 제공할 수 있지만 일반적인 프로덕션 용도의 경우 Kubernetes 클러스터 외부에서 이러한 서비스를 제공하는 것이 좋습니다.

- MySQL 5.7 데이터베이스
  - Amazon RDS MySQL 5.7 또는 MySQL 5.7 데이터베이스(외부)
  - Mysql Bitnami Helm 차트(내부)
  - 파일 스토리지
    - Amazon S3 또는 S3 호환 스토리지 공급자(외부)
    - 미니오 연산자 차트(내부)

네트워크 요구 사항

Wickr Enterprise에는 FQDN, SSL 인증서, 특정 열린 TCP 및 UDP 포트가 필요합니다.

- FQDN: Wickr Enterprise 배포에서 사용할 도메인 또는 하위 도메인입니다.
- SSL 인증서: 퍼블릭 CA 또는 자체 서명된 인증서 키 페어로 서명된 SSL 인증서 키 페어입니다. 인증서는 일반 이름에 FQDN을 나열하고 SAN DNS 항목으로도 나열해야 합니다. 인증서는 serverAuth extendedKeyUsage 확장도 활성화해야 합니다.

- 온라인 설치에는 복제 및 타사 리소스에 대한 외부 액세스 권한이 필요합니다. 복제된는 IP 주소 목록을 유지합니다. 자세한 내용은 복제된 IP 주소를 참조하세요. 또한 복제된는 필요한 타사 리소스 목록을 유지합니다. 자세한 내용은 Firewall Openings for Online Installations를 참조하세요.
- 에어 갭 설치에는 프라이빗 컨테이너 레지스트리에 대한 액세스 권한이 필요합니다.

메시징 노드

메시징 노드에는 퍼블릭 IPV4 주소가 필요하지 않으며 프라이빗 서브넷에 있어야 합니다. 메시지 트래픽은 LoadBalancer 또는 Ingress를 통해 클러스터로 들어갑니다.

노드 호출

호출 노드에는 퍼블릭 IPV4 주소가 필요하므로 퍼블릭 서브넷에 있어야 합니다. 통화 미디어는 기본적으로 UDP를 통해 전송됩니다. TCP 호출이 활성화되면 TCP 프록시는 TCP 443의 연결을 수락하고 이를 Orville 서비스에 프록시합니다.

- TCP: 443 TCP 프록시 호출
- UDP: 16384-16484 오디오/비디오 스트림

설치 및 구성 액세스

설치 및 구성을 위한 KOTS 관리 콘솔에 대한 액세스는 Kubernetes 포트를 통해 수행됩니다.

```
kubectl kots admin-console -n wickr
```

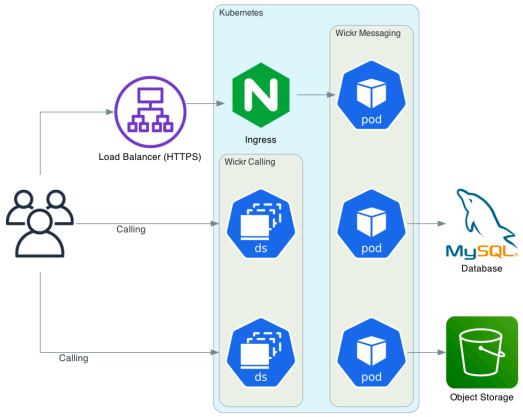
라이선스 요구 사항

설치에는 .yaml 형식의 라이선스 파일이 필요하며, 이는 Wickr Support에서 제공합니다.

아키텍처

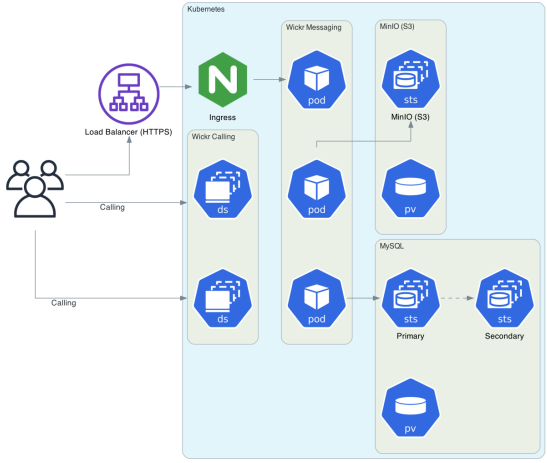
권장 프로덕션 아키텍처

아래 다이어그램은 Kubernetes 클러스터 외부에 MySQL 및 객체 스토리지 서비스가 모두 있는 프로덕션용으로 권장된 대로 구성된 Wickr Enterprise를 보여줍니다.



내부 또는 테스트 아키텍처

아래 다이어그램은 내부 MYSQL 및 객체 스토리지 서비스를 활용하여 Wickr Enterprise의 구성을 보여줍니다. 특정 배포의 특정 요구 사항을 충족할 수 있지만 일반적인 프로덕션 용도로는 권장되지 않습니다.



설치

1. kubectl 및 kots CLI를 설치합니다.

2. Kubernetes 클러스터에 연결합니다.
3. Wickr Support에서 Wickr Enterprise 라이선스 파일을 가져옵니다.
4. 다음 명령을 사용하여 Wickr Enterprise를 설치합니다.

```
kubectl kots install wickr-enterprise-ha \
   --license-file ./license.yaml \
   --namespace wickr
```

> **Note**
>
> license.yaml은 제공된 라이선스 파일을 나타냅니다.

초기 설치 후 KOTS 관리 콘솔은 클러스터 수준 관리 및 구성 옵션을 제공합니다.

KOTS 관리 콘솔

이 인터페이스는 Wickr Enterprise의 배포된 버전을 관리하는 데 사용됩니다. 설치 상태를 확인하거나, 구성을 수정하거나, Wickr Enterprise의 업그레이드를 수행할 수 있습니다. KOTS 관리 콘솔은 다음 명령을 사용해야만 열 수 있는 Kubernetes 포트 포워드를 통해서만 액세스할 수 있습니다.

```
kubectl kots admin-console -n wickr
```

수신 설정

수신 컨트롤러

Wickr Enterprise는 네 가지 수신 컨트롤러 유형을 지원합니다.

- LoadBalancer(기본값)
  - 로드밸런서 객체는 클라우드 공급자가 제공하는 경우가 많더라도 온프레미스 완전 설치에서 명시적 구성이 필요할 수 있습니다.
  - LoadBalancer 서비스 유형을 사용하여 수신 컨트롤러(inress-nginx) 서비스를 배포합니다. 이를 위해서는 Kubernetes 클러스터가 외부 로드 밸런서를 지원하는 플랫폼에서 실행되어야 합니다.
- 기존 ALB
  - 수신 컨트롤러를 기존 ALB에 연결합니다.

  - 기존 Application Load Balancer 대상 그룹 ARN을 제공해야 합니다.
- 기존 NLB
  - 수신 컨트롤러를 기존 NLB에 연결합니다.
  - 기존 Network Load Balancer 대상 그룹 ARN을 제공해야 합니다.
- NodePort
  - 수신 컨트롤러(inress-nginx)는 Kubernetes 클러스터의 모든 노드에서 포트를 열고 트래픽을 수신으로 전달하는 NodePort 서비스 유형을 사용하도록 구성됩니다. 그런 다음 DNS 또는 일부 외부 로드 밸런서를 통해 클라이언트 트래픽을 이러한 노드로 보낼 수 있습니다.
  - 1~65535의 포트 범위를 선택할 수 있습니다. 그렇지 않으면 30000-32767의 임의 포트가 사용됩니다.
- Ingress
  - 자체 수신 컨트롤러를 가져옵니다. 이 구성은 서비스가 수신 매니페스트에 사용할 수신 클래스 이름을 수락합니다. 이는 수신 컨트롤러에 다른 로드 밸런싱 메커니즘을 통해 이미 구성된 외부 연결이 있음을 의미합니다.
  - 현재 ingress-nginx 컨트롤러만 지원됩니다.

와일드카드 호스트 이름

기본적으로 수신 경로는 호스트 값 `*`로 정의됩니다. Wickr Enterprise Server에 대해 정의된 호스트 이름을 사용하려면이 설정을 비활성화합니다. IP 기반 호스트 이름에는 와일드카드 호스트 이름이 필요합니다.

데이터베이스 설정

Wickr Enterprise에는 MySQL 8.0 데이터베이스가 필요합니다. MySQL 5.7을 사용하는 경우를 참조하여 업그레이드MySQL 8.0으로 업그레이드하세요. Amazon RDS와 같이 Kubernetes 클러스터 외부에 있는 데이터베이스를 사용하는 것이 좋지만 설치의 일부로 Kubernetes 클러스터 내부에 내부 MySQL 데이터베이스를 배포하는 옵션도 있습니다.

외부 데이터베이스 설정

- 호스트 이름: 데이터베이스 서버의 호스트 이름 또는 IP 주소입니다.
- 리더 호스트 이름: 데이터베이스 서버에 대한 읽기 전용 엔드포인트의 호스트 이름 또는 IP 주소(사용 가능한 경우).
- 포트: MySQL에 액세스할 포트입니다.

- 데이터베이스 이름: 서버에서 생성된 데이터베이스의 이름입니다.
- 사용자 이름: 데이터베이스에 액세스할 수 있는 권한이 있는 사용자입니다.
- 암호: 해당 사용자의 암호입니다.
- CA 인증서: TLS를 통해 데이터베이스에 연결하기 위한 PEM 인증서입니다.

> **Note**
>
> MySQL 설치에서 latin1_swedish_ci 데이터 정렬과 함께 기본 latin1 문자 세트를 사용하고 있는지 확인합니다. MySQL 서버가 다음 플래그로 시작되었는지 확인하여이 작업을 수행할 수 있습니다.
> `"--character-set-server latin1", "--collation-server latin1_swedish_ci"`

내부 데이터베이스 설정

내부 데이터베이스 유형은 이진 복제를 사용하는 MySQL 기본 및 보조에 대해 클러스터에 StatefulSets 2개를 배포합니다. 보조는 트래픽을 수신하지 않으며 재해 복구 및 백업에만 사용할 수 있습니다.

스토리지 크기: 데이터베이스 포드의 영구 볼륨 크기(기비바이트)입니다.

MySQL 스토리지 크기 늘리기

> **Note**
>
> 스토리지 크기를 늘리려면 StorageClass의 볼륨 유형이 볼륨 확장을 지원해야 합니다. 자세한 내용은 볼륨 확장을 참조하세요.

Wickr Enterprise에서 사용되는 MySQL 서비스는 Kubernetes에서 StatefulSet 리소스로 배포됩니다. StatefulSets는 영구 볼륨 클레임 템플릿을 포함하여 리소스의 많은 속성을 변경할 수 없습니다. StatefulSets의 불변성을 해결하기 위한 차선책으로 MySQL에서 사용하는 볼륨의 크기를 늘리려면 다음 작업을 수행해야 합니다.

1. `data-mysql-primary-0` 및에 대한 영구 볼륨 클레임을 편집합니다`data-mysql-secondary-0`.

1. `kubectl -n wickr edit pvc data-mysql-primary-0.` Set `spec.resources.requests.storage` 원하는 스토리지 크기로 변경합니다.
2. `kubectl -n wickr edit pvc data-mysql-secondary-0.` Set `spec.resources.requests.storage` 원하는 스토리지 크기로 변경합니다.

2. 기존 StatefulSets를 삭제하되 `--cascade=orphan` 플래그를 전달하여 포드를 그대로 둡니다.

   `kubectl -n wickr delete statefulset --cascade=orphan mysql-primary mysql-secondary`.

3. KOTS UI에서 스토리지 크기 설정을 1단계에서 설정한 값과 일치하도록 업데이트합니다. 이 구성을 저장하고 배포합니다.
4. StatefulSets를 다시 시작하여 볼륨을 확장하고 MySQL 서비스를 다시 온라인 상태로 전환합니다.

   `kubectl -n wickr rollout restart statefulset mysql-primary mysql-secondary`.

MySQL 8.0으로 업그레이드

외부 데이터베이스(RDS)

Wickr 백엔드를 오프라인으로 전환하려면 다음 단계를 완료합니다.

1. 수신 네임스페이스 찾기 `kubectl get deployments --all-namespaces`

   아래 예제에서 네임스페이스는 Wickr이고 복제본은 3입니다.

```
NAMESPACE     NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
...
wickr         ingress-nginx-controller   3/3     3            3           43h
...
```

2. 수신 축소 `kubectl scale deployment/ingress-nginx-controller --replicas=0 -n wickr`
3. 스냅샷을 생성하여 DB를 백업합니다. 자세한 내용은 Amazon Relational Database Service 사용 설명서의 수동 백업 관리를 참조하세요.
4. 엔진 버전을 MySQL 8.0.x로 업그레이드합니다(MySQL 8.4는 지원되지 않음). 자세한 내용은 Amazon Relational Database Service 사용 설명서의 DB 인스턴스 엔진 버전 업그레이드를 참조하세요.

Wickr 백엔드를 온라인 상태로 전환하려면 수신을 축소합니다. `kubectl scale deployment/ingress-nginx-controller --replicas=3 -n wickr`

내부 데이터베이스

자세한 내용은 MySQL 백업 및 복원을 참조하세요.

S3 파일 스토리지

Wickr Enterprise에는 S3 호환 스토리지 서비스가 필요합니다. Amazon S3와 같이 Kubernetes 클러스터 외부에 있는 S3 서비스를 사용하는 것이 좋지만 설치의 일부로 Kubernetes 클러스터 내부에 내부 S3 서비스를 배포할 수도 있습니다. Amazon S3

외부 S3 설정

- 버킷 이름: 파일 업로드가 저장될 S3 버킷의 이름입니다.
- 리전: S3 버킷의 AWS 리전입니다.
- 엔드포인트: Wickr가 S3 API와 상호 작용하는 데 사용할 엔드포인트를 설정합니다. 기본값은 리전의 S3 서비스 엔드포인트입니다.
- Fileproxy 서비스 계정 이름: Amazon S3 전용. 서비스 계정에 대한 IAM 역할을 사용하여 S3에 인증하는 데 사용할 기존 Kubernetes 서비스 계정의 이름입니다.
- 외부 S3 액세스 키: 기존 S3 액세스 키입니다.
- 외부 S3 비밀 키: 기존 S3 비밀 키입니다.

내부 S3 설정

내부 S3 유형은 각각 4개의 영구 볼륨 클레임을 포함하는 4개의 MinIO 서버 포드 기본값을 배포합니다. 기본 구성은 MinIO의 삭제 코딩을 활용하여 내결함성을 높입니다.

- 내부 S3 서버 수: 생성할 MinIO 서버 포드 수로, 내결함성 배포의 기본값은 4입니다. 개발/테스트 배포의 경우이 값을 1로 낮게 설정할 수 있습니다.
- 내부 S3 볼륨 수: 각 MinIO 서버 포드에서 생성할 MinIO 볼륨 수입니다. 내결함성 배포의 기본값은 4입니다. 개발/테스트 배포의 경우이 값을 1로 낮게 설정할 수 있습니다.
- 내부 S3 볼륨 크기: MinIO 서버 포드에서 생성된 MinIO 볼륨의 GB 크기로, 기본값은 10GB입니다.
- 기본 내부 S3 배포는 PVC가 4개인 서버 4PVCs 사용합니다. 각 PVC는 10Gi로, 사용자가 사용할 수 있는 120Gi Erasure Coded 스토리지와 함께 160Gi Raw 스토리지를 생성합니다.

- Minio Erasure Coding 계산기를 사용할 수 있습니다. 자세한 내용은 코드 계산기 삭제를 참조하세요.

영구 볼륨 클레임 설정

Wickr Enterprise가 상태 저장 데이터를 저장하려면 영구 볼륨 클레임이 필요합니다. 이 설정을 사용하면 사용하려는 스토리지 클래스의 이름 이름을 지정할 수 있습니다. 비워 두면 Wickr는 기본 스토리지 클래스를 사용하려고 시도합니다. Wickr 배포 후 스토리지 클래스 변경은 지원되지 않습니다.

영구 볼륨 클레임에 대한 기본 StorageClass는 클라우드 공급자가 제공하는 경우가 많지만 완전 온프레미스 설치에서는 Longhorn과 같은 타사 서비스를 사용하여 명시적으로 구성해야 할 수 있습니다.

TLS 인증서 설정

TLS를 종료하기 위한 PEM 인증서와 프라이빗 키를 업로드합니다. 인증서의 주체 대체 이름은 Wickr Enterprise 배포의 설정에 구성된 호스트 이름과 일치해야 합니다.

인증서 체인 필드의 경우 업로드하기 전에 중간 인증서(필요한 경우)를 루트 CA 인증서와 연결합니다.

Let's Encrypt

Let's Encrypt를 사용하여 인증서를 자동으로 생성하려면이 옵션을 선택합니다. 인증서는 cert-manager 연산자를 통해 HTTP-01 챌린지를 사용하여 발급됩니다.

HTTP-01 챌린지에서는 원하는 DNS 이름이 클러스터의 수신 지점(일반적으로 Load Balancer서)으로 확인되고 TCP 포트 80에 대한 트래픽이 퍼블릭에 열려 있어야 합니다. 이러한 인증서는 수명이 짧으며 정기적으로 갱신됩니다. 인증서를 자동으로 갱신하려면 포트 80을 열어 두어야 합니다.

> **Note**
>
> 이 섹션에서는 Wickr Enterprise 애플리케이션 자체에서 사용하는 인증서를 명시적으로 참조합니다.

고정된 인증서

Wickr Enterprise는 자체 서명된 인증서 또는 클라이언트 디바이스에서 신뢰할 수 없는 인증서를 사용할 때 인증서 고정이 필요합니다. Load Balancer서에서 제공하는 인증서가 자체 서명되었거나 Wickr Enterprise 설치와 다른 CA에서 서명한 경우 여기에 CA 인증서를 업로드하여 클라이언트가 대신 인증서를 고정하도록 합니다.

대부분의 경우이 설정은 필요하지 않습니다.

인증서 공급자

Wickr Enterprise에서 사용할 인증서를 구매하려는 경우 인증서가 기본적으로 올바르게 작동하는 것으로 알려진 공급자 목록은 아래를 참조하세요. 공급자가 아래에 나열된 경우 해당 인증서는 소프트웨어를 통해 명시적으로 검증되었습니다.

- 다이제르트
- RapidSSL

자체 서명된 인증서 생성

Wickr Enterprise에서 사용할 자체 서명된 인증서를 생성하려면 아래 예제 명령에 생성에 필요한 모든 플래그가 포함되어 있습니다.

```
openssl req -x509 -newkey rsa:4096 -sha256 -days 365 -nodes -keyout $YOUR_DOMAIN.key -
out $YOUR_DOMAIN.crt -subj "/CN=$YOUR_DOMAIN" -addext "subjectAltName=DNS:$YOUR_DOMAIN"
 -addext "extendedKeyUsage = serverAuth"
```

IP 기반 자체 서명 인증서를 생성하려면 대신 다음 명령을 사용합니다. IP 기반 인증서를 사용하려면 수신 설정에서 와일드카드 호스트 이름 필드가 활성화되어 있는지 확인합니다. 자세한 내용은 수신 설정을 참조하세요.

```
openssl req -x509 -newkey rsa:4096 -sha256 -days 365 -nodes -keyout $YOUR_DOMAIN.key -
out $YOUR_DOMAIN.crt -subj "/CN=$YOUR_DOMAIN" -addext "subjectAltName=IP:$YOUR_DOMAIN"
 -addext "extendedKeyUsage = serverAuth"
```

> **Note**
>
> 예제의 $YOUR_DOMAIN을 사용하려는 도메인 이름 또는 IP 주소로 바꿉니다.

호출 설정

- 호출 노드 필요:이 설정이 활성화되면 Wickr의 호출 서비스는 레이블이 인 Kubernetes 노드에만 배포됩니다`role=calling`. 동일한 노드에 Calling 및 Messaging 서비스를 배포하거나 단일 노드 배포를 위해이 설정을 비활성화합니다.

TCP 프록시 서비스는 포트 443에서 실행되므로 일반적으로이 설정을 비활성화할 때 호출 TCP 프록시를 비활성화해야 합니다.

- TCP 프록시 활성화:이 설정은 호출 시 TCP 폴백 모드에 대한 서비스가 배포되는지 여부를 제어합니다. 443/tcp에서 실행 중인 다른 서비스가 있거나 호출에 TCP 폴백 모드가 필요하지 않은 경우이 설정을 비활성화합니다. Wickr Open Access를 사용하려는 배포에 대해 활성화해야 합니다.
- 서버 퍼블릭 IP 주소 자동 검색:이 설정이 활성화되면 호출 서비스는 https://ipv4.icanhazip.com/ 및 에 HTTPS를 요청하여 퍼블릭 IP 주소를 검색합니다https://ipv6.icanhazip.com/. 비활성화된 경우 “트래픽 호출에 호스트 기본 IP 주소 사용” 또는 “호스트 이름 재정의” 설정을 활성화해야 합니다. 그렇지 않으면 호출 서비스가 시작되지 않습니다.
- 트래픽 호출에 호스트 기본 IP 주소 사용: 서비스 호출에 Kubernetes 노드의 기본 IP 주소를 사용합니다. 이는 다운워드 API`status.hostIP`의에 표시된 대로 모든 Wickr 클라이언트가 노드의 기본 IP 주소에 있는 Kubernetes 노드에 연결할 수 있음을 의미합니다.
- 호스트 이름 재정의: 호출 서비스의 연결 지점으로 반환할 호스트 이름 또는 IP 주소를 제공합니다. 이 설정은 단일 호출 서버를 실행할 때만 사용해야 합니다. 서비스의 모든 복제본에 대해 동일한 값이 반환되기 때문입니다. 호스트 이름 재정의가 설정되고 '호스트 기본 IP 주소 사용' 설정이 활성화된 경우 호스트 기본 IP 주소 설정이 우선합니다.
- 호스트 네트워크 호출 활성화: 기본적으로 포드 호출은 연결을 위해 노드의 호스트 네트워크를 사용합니다. 트래픽을 호출하기 위한 NodePort 서비스를 노출하려면이 기능을 비활성화합니다. 수신 호출이 활성화된 경우 수신 트래픽을 허용하도록 적절한 서비스가 구성되어 있는지 확인합니다. STIG 규정 준수를 위해 비활성화해야 합니다.

수신 설정 호출

Wickr는 호출 수신 설정을 지원하므로 클라이언트가 클러스터 내의 모든 호출 노드에 연결하고 호출 경로를 올바른 호출 서버로 지정할 수 있습니다. Wickr는 네 가지 호출 수신 유형을 지원합니다.

- LoadBalancer(기본값)
  - LoadBalancer는 클라우드 공급자가 프로비저닝합니다(완전한 온프레미스 설치에는 추가 구성이 필요함). LoadBalancer가 프로비저닝된 후 로드 밸런서의 호스트 이름 또는 IP 주소를 제공하도록 KOTS 구성을 다시 업데이트해야 합니다.
- NodePort
  - 트래픽을 호출하기 위한 진입점 역할을 할 각 호출 노드에서 NodePort 서비스를 노출합니다. 하나 이상의 노드로 확인되는 호스트 이름 또는 하나 이상의 노드의 IP 주소를 제공해야 합니다. UDP 및 선택적으로 TCP 트래픽에 대해 30000-32767의 포트 범위를 선택할 수 있습니다.

- 기존 NLB
  - 호출 수신 서비스를 기존 NLB에 연결합니다. UDP 및 선택적으로 TCP 트래픽에 대상 그룹 ARN을 제공해야 합니다.
- 서비스 없음
  - 수신 트래픽을 허용하는 데 추가 Kubernetes 서비스가 필요하지 않은 경우이 옵션을 선택합니다. 일반적으로 호스트 네트워크 설정과 함께 사용하여 호출 수신 트래픽을 호출 노드로 직접 라우팅합니다.

고려 사항

- 수신을 호출하지 않고 이전 클라이언트 및 페더레이션 네트워크와의 이전 버전과의 호환성을 보장하기 위해 수신을 호출할 때 레거시 호출 모드를 계속 사용할 수 있습니다(호출 서버에 직접 연결). 기본 포트를 변경하는 경우 호출 노드에 포트 충돌이 없는지 확인합니다.
- UDP 트래픽을 제공하는 듀얼 스택 NLBs에는 IPv6 백엔드 대상이 있어야 합니다. 자세한 내용은 Network Load Balancer 대상 그룹을 참조하세요.
- STIG 규정 준수가 필요한 경우 호출을 위한 호스트 네트워크 옵션을 비활성화해야 합니다. 노드가 듀얼 스택 모드로 구성되었지만 클러스터가 구성되지 않은 경우 IPv6 연결이 끊어질 수 있습니다(IPv4 클러스터 가정).
- 수신을 호출하려면 미리 정의된 호스트 이름 또는 IP 주소가 필요합니다. 노드를 조정하거나 사용자 지정 라우팅을 제공하려면 구성을 수정해야 할 수 있습니다.
- 기본 호출 수신 포트는 TCP의 경우 8443, UDP의 경우 16384입니다. 방화벽과 보안 그룹이 이러한 포트 또는 기본값이 재정의된 경우 대체 포트에 대한 트래픽을 허용하는지 확인합니다.

참조 아키텍처

로드 밸런서로 수신

이 옵션은 단일 로드 밸런서를 모든 호출 트래픽의 진입점으로 노출합니다.

1. 호출 수신 유형에서 Load Balancer 또는 기존 NLB를 선택합니다. 기존 NLB에 대한 자세한 내용은 GitHub의 Wickr Enterprise CDK 샘플에서 NLB 스택을 참조하세요.
2. 호출 수신 유형에 따라 다음 중 하나를 수행합니다.
   - 기존 NLB의 경우 UDP 및 TCP 트래픽의 대상 그룹 ARNs과 NLB의 호스트 이름을 제공합니다.
   - Load Balancer의 경우 Kubernetes에서 프로비저닝한 후 호스트 이름을 제공합니다.

또는 호출 수신 유형에서 로드 밸런서의 IP 주소 또는 로드 밸런서를 가리키는 사용자 지정 호스트 이름을 제공할 수 있습니다.

3. (선택 사항) 단일 NLB에서 메시징 및 호출 트래픽을 결합하려면 수신 섹션에서 기존 NLB를 선택하고 HTTPS 대상 그룹을 제공합니다.

NodePort로 수신

이 옵션은 호스트 네트워킹이 비활성화되어 있고 추가 로드 밸런서를 노출하지 않으려는 경우에 유용합니다.

> **Note**
> 방화벽과 보안 그룹이 NodePorts 대한 트래픽을 허용하는지 확인합니다.

1. 수신 유형 호출에서 NodePort를 선택합니다.
2. 호출 노드 호스트 이름 또는 IP 주소를 추가합니다.
3. 호스트 네트워크 호출을 비활성화합니다.

HostNetwork로 직접 수신

이 옵션은 추가 Kubernetes 서비스를 노출하지 않으며 호출 수신 트래픽이 호출 노드의 호스트 네트워크를 통해 직접 연결되도록 허용합니다. IPv6 연결이 필요한 경우이 접근 방식을 사용하는 것이 좋습니다.

1. 수신 유형 호출에서 서비스 없음을 선택합니다.
2. 호출 노드 호스트 이름 또는 IP 주소를 추가합니다.
3. 호스트 네트워크 호출을 활성화합니다.

Kubernetes 클러스터 오토스케일러(선택 사항)

Kubernetes Cluster Autoscaler는 Wickr Enterprise 설치를 위한 선택적 구성 값입니다. 트래픽 증가 또는 성능 저하로 이어질 수 있는 기타 리소스 제한 발생 시 Kubernetes 노드 그룹을 조정하는 데 도움이 됩니다.

Wickr Enterprise 설치는 3가지 클라우드 공급자 통합 AWS, 즉 Google Cloud 및 Azure를 지원합니다. 클라우드 공급자마다이 통합에 대한 요구 사항이 다릅니다. 이 기능을 활성화하려면 아래 특정 클라우드 공급자의 지침을 따르세요.

AWS

WickrEnterpriseCDK를 사용하여 Wickr 환경을 설치하지 않은 경우 Cluster Autoscaler를 활성화하기 위한 몇 가지 추가 단계를 수행해야 AWS합니다.

1. 노드 그룹에 다음 태그를 추가합니다. 이렇게 하면 Cluster Autoscaler가 적절한 노드를 자동으로 검색할 수 있습니다.

    1. `k8s.io/cluster-autoscaler/`**`clusterName`** `= owned` 여기서 clusterName은 Kubernetes 클러스터의 이름입니다.
    2. `k8s.io/cluster-autoscaler-enabled = true`

2. kube-system 네임스페이스에 Kubernetes 서비스 계정을 추가하고 Auto Scaling 및 ec2 작업을 허용하는 IAM 정책과 연결합니다. 자세한 내용과 자세한 지침은 Amazon EKS 사용 설명서의 IAM 역할을 수임하도록 Kubernetes 서비스 계정 구성을 참조하세요.

    1. 서비스 계정을 설정할 때 'kube-system' 네임스페이스를 사용해야 합니다.
    2. 서비스 계정에는 다음 정책을 사용할 수 있습니다.

        JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:DescribeAutoScalingInstances",
                "autoscaling:DescribeLaunchConfigurations",
                "autoscaling:DescribeTags",
                "autoscaling:SetDesiredCapacity",
                "autoscaling:TerminateInstanceInAutoScalingGroup",
                "ec2:DescribeInstanceTypes",
                "ec2:DescribeInstances",
                "ec2:DescribeLaunchTemplateVersions"
               ],
            "Resource": "*",
            "Effect": "Allow"
```

```
        }
    ]
}
```

Cluster Autoscaler를 구성할 때 복제된 UI에서 클라우드 공급자AWS로를 선택하고 위에서 생성한 서비스 계정의 이름을 제공하여 Cluster Autoscaler에 해당 서비스 계정을 활용하도록 지시합니다.

Google 클라우드

Autopilot 및 표준 클러스터 모두에 대해 GKE의 내장 Autoscaling 기능을 사용하는 것이 좋습니다. 그러나이 통합을 진행하려면 진행하기 전에 다음 요구 사항을 충족해야 합니다.

요구 사항:

1. 관리형 인스턴스 그룹(MIG)은 컴퓨팅 엔진 리소스에 대한 최소 '읽기/쓰기'를 포함하여 보안 범위를 사용하여 생성해야 합니다. 이는 현재 나중에 MIG에 추가할 수 없습니다.
2. 클러스터에는 워크로드 자격 증명 연동이 활성화되어 있어야 합니다. 다음을 실행하여 기존 클러스터에서이 기능을 활성화할 수 있습니다. `gcloud container clusters update ${CLUSTER_NAME} --workload-pool=${PROJECT_ID}.svc.id.goog`
3. `roles/compute.instanceAdmin.v1`. 다음 지침을 사용하여 생성할 수 있습니다.

```
# Create GCP Service Account
gcloud iam service-accounts create k8s-cluster-autoscaler

# Add role to GCP Service Account
gcloud projects add-iam-policy-binding ${PROJECT_ID} \
--member "serviceAccount:k8s-cluster-autoscaler@${PROJECT_ID}.iam.gserviceaccount.com"
 \
--role "roles/compute.instanceAdmin.v1"

# Link GCP Service Account to Kubernetes Service Account
gcloud iam service-accounts add-iam-policy-binding k8s-cluster-autoscaler@
${PROJECT_ID}.iam.gserviceaccount.com \
--role roles/iam.workloadIdentityUser \
--member "serviceAccount:${PROJECT_ID}.svc.id.goog[kube-system/cluster-autoscaler-gce-
cluster-autoscaler]"
```

Azure

Azure Kubernetes Service(AKS)는 대부분의 배포에 통합 클러스터 Autoscaling을 제공하며 클러스터 Autoscaling에 이러한 방법을 사용하는 것이 좋습니다. 그러나 요구 사항이 해당 메서드가 작동하지 않는 경우 Azure Kubernetes Service에 대한 Kubernetes Cluster Autoscaler 통합을 제공했습니다. 이 통합을 활용하려면 다음 정보를 수집하고 Azure를 클라우드 공급자로 선택한 후 Cluster Autoscaler 아래의 KOTS 관리자 패널 구성에 넣어야 합니다.

Azure 인증

구독 ID: 공식 설명서에 따라 Azure 포털을 통해 구독 ID를 가져올 수 있습니다. 자세한 내용은 Azure 포털의 구독 및 테넌트 IDs 가져오기를 참조하세요.

az 명령줄 유틸리티를 사용하여 AD 서비스 보안 주체를 생성하여 다음 파라미터를 가져올 수 있습니다.

```
az ad sp create-for-rbac –role="Contributor" –scopes="/subscriptions/subscription-id" –
output json
```

앱 ID:

클라이언트 암호:

테넌트 ID:

Azure Cluster Autoscaler 구성

인증 요구 사항 외에도 클러스터 오토스케일러가 제대로 작동하려면 다음 필드가 필요합니다. 이 정보를 얻기 위한 명령은 편의를 위해 제공되었지만 특정 AKS 구성에 따라 일부 수정이 필요할 수 있습니다.

Azure 관리형 노드 리소스 그룹:이 값은 정의한 리소스 그룹이 아닌 AKS 클러스터를 설정할 때 Azure에서 생성한 관리형 리소스 그룹입니다. 이 값을 얻으려면 클러스터를 생성할 때의 CLUSTER_NAME 및 RESOURCE_GROUP이 필요합니다. 이러한 값이 있으면 다음을 실행하여이 값을 얻을 수 있습니다.

```
az aks show –resource-group ${RESOURCE_GROUP} –name ${CLUSTER_NAME} –query
 nodeResourceGroup -o tsv
```

애플리케이션 노드 풀 VMSS 이름: Wickr 애플리케이션의 AKS 노드 풀과 연결된 가상 머신 조정 세트(VMSS)의 이름입니다. 클러스터의 요구 사항에 따라 확장 또는 축소되는 리소스입니다. 이 값을 얻으려면 다음 az 명령을 실행할 수 있습니다.

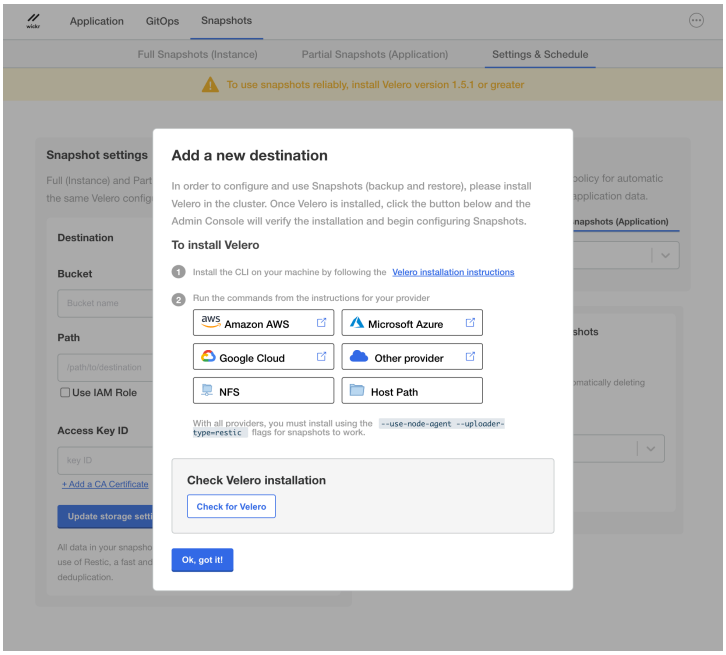
```
CLUSTER_NODEPOOL_NAME="(Your-NodePool-Name)"
CLUSTER_RESOURCE_GROUP="(Your-Managed-Node-Resource-Group-As-Defined-Above>)"
az vmss list -g ${CLUSTER_RESOURCE_GROUP} --query '[?tags."aks-managed-
poolName"==`'''${CLUSTER_NODEPOOL_NAME}'''`].{VMSS_name:name}' -o tsv
```

ACalling 노드 풀 VMSS 이름(선택 사항): 노드 풀이 있는 경우 호출 노드 풀과 연결된 VMSS의 이름입니다. 이 값을 얻으려면 애플리케이션 노드 풀 VMSS 이름에 대해 수정된 버전의 명령을 실행하여 호출 노드 풀의 노드 풀 이름에 대한 CLUSTER_NODEPOOL_NAME 값을 전환할 수 있습니다.

백업

Wickr Enterprise는 백업 목적으로 Velero를 활용합니다. Velero는 클라우드 공급자에서 운영하든 온프레미스에서 운영하든 관계없이 Kubernetes 클러스터 리소스 및 영구 볼륨을 백업하고 복원하는 데 필요한 도구를 제공합니다.

미니오를 사용한 Velero 백업: 현재 Velero 백업은 리소스 부족 모드에서 미니오에 대해서만 활성화됩니다.



Velero 설명서를 사용한 설치

- Velero CLI를 설치합니다. 자세한 내용은 Velero CLI 설치를 참조하세요.
- 클러스터에 Velero를 설치하고 공급자를 기반으로 스토리지를 구성합니다.
  - AWS.
  - GCP.
  - Azure.
  - 기타 공급자.

제한 사항

기본적으로 백업에는 볼륨이 포함되지 않습니다. 포드가 백업해야 하는 볼륨을 탑재하는 경우 백업에 포함할 특정 볼륨을 나열하는 주석으로 백업을 구성해야 합니다.

백업이 필요한 각 볼륨에 대해 backup.velero.io/backup-volumes 주석을 추가합니다. 주석 이름은 backup.velero.io/backup-volumes, 값은 백업에 포함할 쉼표로 구분된 볼륨 목록입니다. 자세한 내용은 스냅샷 구성을 참조하세요.

Airgap 설치

Wickr Enterprise와 KOTS는 모두 완전히 에어 갭된 Kubernetes 클러스터로의 배포를 지원합니다. 에어갭 Kubernetes 클러스터에서 연결할 수 있는 프라이빗 도커 이미지 레지스트리에 대한 액세스를 제공해야 합니다. 이 용도로 올바르게 작동하려면 KOTS에 제공된 Private Docker Image Registry를 사용자 이름/암호 인증으로 보호해야 합니다. KOTS는 프라이빗 도커 이미지 레지스트리를 활용하여 모든 Wickr Enterprise 이미지를 호스팅합니다.

- airgap이 활성화된 Wickr Enterprise license.yaml(Wickr 영업 또는 고객 지원 팀에 문의)
- Wickr Enterprise wickr.airgap 아카이브 번들(Wickr 영업 또는 고객 지원 팀에 문의)
- 프라이빗 도커 이미지 레지스트리에 대한 액세스.
- Airgap 환경에 배포된 Kubernetes 클러스터에 대한 액세스입니다.
- Kubectl이 설치되었습니다.
- KOTS CLI가 설치되었습니다.
- kotsadm.tar.gz가 다운로드되었습니다.

다음 명령을 실행하여 에어갭 kubernetes 클러스터에 KOTS 및 Wickr Enterprise를 배포합니다. 이 명령은 KOTS 관리자 이미지와 Wickr Enterprise 이미지를 Private Docker Image Registry에 업로드합니다. 명령이 완료되면 위와 같이 KOTS 관리 콘솔에 액세스하여 Wickr Enterprise 설치를 완료하라는 메시지가 표시됩니다.

```
kubectl kots admin-console push-images \
  ~/kotsadm.tar.gz $PRIVATE_REGISTRY_HOST \
  --registry-username $PRIVATE_REGISTRY_USER \
  --registry-password $PRIVATE_REGISTRY_PASSWORD

kubectl kots install wickr \
  --license-file ~/YOUR_LICENSE.yaml \
  --airgap-bundle ~/wickr.airgap \
  --kotsadm-registry $PRIVATE_REGISTRY_HOST \
  --registry-username $PRIVATE_REGISTRY_USER \
  --registry-password $PRIVATE_REGISTRY_PASSWORD
```

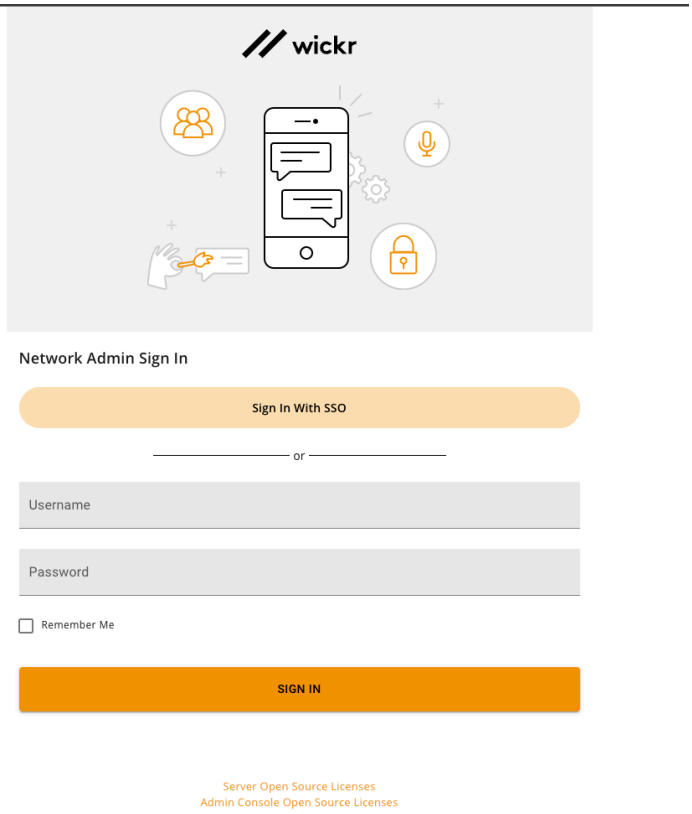
에어갭 설치를 위한 모바일 알림

서버 백엔드에서 모바일 클라이언트로 푸시 알림을 보내려면 추가 네트워킹 허용 목록이 필요합니다. 이 요구 사항은 Apple iOS 및 Google Android가 오프라인 및 백그라운드 디바이스에 대해이 기능을 구현하는 방식 때문입니다. 이러한 서비스에 대한 설명서를 참조하고 지정된 IP 주소 및 포트를 허용 목록으로 표시합니다.

- iOS
- Android

Wickr 관리자 콘솔

Wickr Admin Console 인터페이스는 Wickr Enterprise 애플리케이션 자체를 관리하는 데 사용됩니다. 네트워크, 사용자, 페더레이션 등을 설정하는 데 사용할 수 있습니다. 로드 밸런서를 가리키도록 구성한 DNS 이름으로 HTTPS를 통해 액세스할 수 있습니다. 기본 사용자 이름은 admin이며 암호는 Password123입니다. 처음 로그인할 때 이 비밀번호를 변경해야 합니다.



보안 설정

AWS Wickr Enterprise는 배포에 향상된 보안 컨텍스트를 적용하기 위한 구성 설정을 제공합니다. 이 더 높은 보안 표준은 포드 및 컨테이너 수준에서 적용되며 보안 기술 구현 가이드(STIG)를 준수하는 데 필요합니다.

향상된 보안 컨텍스트를 적용하려면 다음 구성 파라미터를 설정합니다.

```
podSecurityContext:
  runAsNonRoot: true
  seccompProfile:
    type: RuntimeDefault
containerSecurityContext:
  allowPrivilegeEscalation: false
  capabilities:
    drop: ["ALL"]
```

> **Warning**
>
> Opensearch의 경우이 보안 구성은 영구 스토리지에 대한 권한을 업데이트하는 `fsgroup-volume` initContainer를 비활성화하므로 권한과 관련된 호환성 문제가 발생할 수 있습니다.

FAQ

Q: helm stderr에서 다음 오류와 함께 배포가 실패합니다.

```
Error: UPGRADE FAILED: cannot patch "enterprise-init" with kind Job:
Job.batch "enterprise-init" is invalid: spec.template: Invalid value: core.
```

A: 디버그 로깅이 활성화된 경우이 문제가 발생할 수 있습니다. 디버그 로깅을 비활성화하고 문제가 있는 작업을 삭제한 다음 다시 시도하세요.

Wickr Enterprise용 임베디드 클러스터

Wickr Enterprise용 임베디드 클러스터 설치 옵션은 Wickr Enterprise 제품을 위한 작고 효율적인 설치 제품을 제공합니다. 복제된 임베디드 클러스터를 활용하여 Wickr Enterprise를 설치할 수 있는 k0을 사용하여 소규모 Kubernetes 설치를 제공합니다. 이 설치 방법을 사용하면 복원력과 고가용성 비용으로 'all-in-one' 솔루션을 제공하여 Wickr Enterprise 설치에 대한 기술 기술 요구 사항과 전체 하드웨어 요구 사항을 최소화할 수 있습니다.

주제

Wickr Enterprise 임베디드 클러스터 시작하기

Wickr Enterprise 임베디드 클러스터 옵션 사용을 시작하려면 지원 센터에 문의하여 라이선스를 받으세요. 기존 라이선스가 있고이 옵션을 활용하려면 지원팀에 문의하여 기존 라이선스 업데이트와 추가 설치 지침을 문의하세요.

Wickr Enterprise 임베디드 클러스터 요구 사항

Wickr Enterprise 임베디드 클러스터를 설치하기 전에 다음 요구 사항이 충족되는지 확인합니다.

네트워크 요구 사항

다음 포트에서 Wickr 서버로의 수신을 허용해야 합니다.

- HTTPS용 443/TCP
- TCP 프록시만 호출 - KOTS에서 TCP 호출 트래픽에 대해 구성된 TCP 프록시 포트
- UDP 호출 트래픽의 경우 16384-19999/UDP
- LAN 전용 - KOTS 관리 콘솔에 액세스하기 위한 30000/TCP

시스템 요구 사항

설치하기 전에 다음과 같은 최소 리소스를 사용할 수 있는 Linux 기반 운영 체제(OS)를 실행하는 VM(가상 머신) 또는 물리적 머신이 있는지 확인합니다.

- CPU 코어 8개
- 12기가바이트(GB)의 RAM
- /(루트) 파티션의 100기가바이트(GB) 디스크 스토리지

Wickr Enterprise 임베디드 클러스터는 다음 Linux OS 시스템에서 테스트되었지만 다른 Linux 기반 OS 옵션도 적합할 수 있습니다.

- Red Hat Enterprise Linux 9.5
- Amazon Linux 2023
- Rocky Linux 9.5

Wickr Enterprise 임베디드 클러스터 설치(표준)

다운로드 지침을 받으면 Wickr Enterprise 번들을 대상 시스템에 다운로드하고 압축을 풉니다.

```
curl -f "https://replicated.app/embedded/wickr-enterprise-ha/stable/6.52" -H
 "Authorization: [redacted]" -o wickr-enterprise-ha-stable.tgz
tar xvf wickr-enterprise-ha-stable.tgz
```

이제 `wickr-enterprise-ha` 및 라는 두 개의 파일이 있어야 합니다`license.yaml`. `wickr-enterprise-ha` 파일은 임베디드 클러스터 설치에 필요한 모든 부분이 포함된 바이너리 파일인 반면, `license.yaml`는 설치를 검증하는 데 사용할 Wickr 라이선스입니다.

이 단계에서 `wickr-enterprise-ha` 파일을 실행하여 기본 설치를 수행할 수 있습니다.

```
./wickr-enterprise-ha install --license license.yaml
```

설치 프로세스가 시작되면 관리자 콘솔 암호를 입력하라는 메시지가 표시됩니다. 설치 구성을 계속하려면 보안 암호를 입력하고 KOTS 관리 콘솔에 액세스할 때 필요한 만큼 암호를 저장해야 합니다.

설치가 완료되면 출력은 다음과 유사합니다.

```
   sudo ./wickr-enterprise-ha install --license license.yaml
? Set the Admin Console password (minimum 6 characters): *********
? Confirm the Admin Console password: *********
# Host files materialized!
# Host preflights succeeded!
# Node installation finished!
# Storage is ready!
# Embedded Cluster Operator is ready!
# Registry is ready!
# Application images are ready!
# Admin Console is ready!
Visit the Admin Console to configure and install wickr-enterprise-ha:
 http://192.168.1.100:30000
```

표준 설치 후 웹 브라우저를 사용하여 출력에 제공된 KOTS 관리자 콘솔 URL로 이동합니다. 이 예제에서 URL은 입니다`http://192.168.1.100:30000`. 그러나 URL은 네트워킹 구성에 따라 달라집니다.

다중 노드 설치

Wickr Enterprise Embedded Cluster 다중 노드 설치는 임베디드 클러스터 사용자가의 Wickr Calling 및 Wickr Messaging 워크로드를 다른 물리적 머신으로 분리할 수 있는 옵션을 제공합니다. 이를 위해 Wickr Enterprise는 복제된 임베디드 클러스터 다중 노드 도구를 활용합니다.

포트 요구 사항

다중 노드 기능이 올바르게 작동하려면 클러스터의 모든 멤버에서 다음 포트가 열려 있어야 합니다. 노드 간에만 열려 있어야 하며 더 넓은 인터넷에서는 열려 있지 않아야 합니다.

- 53 TCP/UDP
- 2380/TCP
- 4789/UDP
- 6443/TCP
- 8080/TCP
- 9091/TCP

- 9443/TCP
- 10249/TCP
- 10250/TCP
- 10256/TCP
- 30000/TCP
- 50000/TCP

라이선스 요구 사항

Wickr 임베디드 클러스터 다중 노드 구성 옵션에는 추가 라이선스 권한이 필요합니다. 지원 에 문의하여 라이선스가이 기능을 지원하는지 확인합니다.

초기 설정 중 추가 노드 생성

Wickr Enterprise Embedded Cluster를 처음 구성할 때 설정 프로세스 중에 추가 호출 노드를 생성할 수 있습니다. 먼저 Wickr Enterprise 임베디드 클러스터 설치(표준)에 설명된 절차를 따릅니다. KOTS 관리자 패널로 이동하면 추가 노드를 생성하라는 메시지가 표시됩니다.

> **Note**
>
> 현재 임베디드 클러스터 다중 노드는 호출 노드 1개와 메시징/컨트롤러 노드 1개만 지원합니다.

시작하려면 컨트롤러 역할 옵션을 선택 취소하고 역할 호출 옵션을 선택합니다. 그러면 새 노드를 구성하기 위한 추가 명령 세트가 채워집니다. 새 노드에서 다음 지침을 실행하여 클러스터를 호출 노드로 조인하도록 구성합니다.

새 노드에서 다음 예제와 유사한 지침을 실행합니다.

1. 새 노드에서 바이너리를 다운로드합니다.

```
curl -k https://172.31.42.64:30000/api/v1/embedded-cluster/binary -o wickr-enterprise-ha.tgz
```

2. 바이너리 추출:

```
tar -xvf wickr-enterprise-ha.tgz
```

3. 노드를 클러스터에 조인합니다.

```
sudo ./wickr-enterprise-ha join 172.31.42.64:30000 AAAAAbbbbbbbbCCCCCCCzzzzz
```

조인 명령이 성공적으로 완료되면 호출 역할이 할당된 클러스터 구성 페이지에 새 노드가 나타납니다. 계속을 선택하여 Wickr Enterprise 구성 페이지로 이동합니다. KOTS 관리자 콘솔 구성에 설명된 임베디드 노드 구성 옵션에 대한 지침을 따릅니다.

기존 임베디드 클러스터 설치에 노드 추가

기존 Wickr Enterprise Embedded Cluster 설치에 호출 노드를 추가하려면 KOTS 관리 콘솔로 이동합니다. 이렇게 하려면 ssh 또는 기타 메커니즘을 통해 노드에 로그인하고 설치에 사용되는 `wickr-enterprise-ha` 바이너리가 포함된 설치 디렉터리로 이동합니다. 를 실행`./wickr-enterprise-ha admin-console`하여 KOTS 관리 콘솔을 시작합니다. 이 명령이 출력을 반환하지 않으면 KOTS 관리 콘솔이 이미 실행 중이며 웹 브라우저의 노드 IP에서 포트 30000으로 이동하여 액세스할 수 있습니다. 예: `https://127.0.0.1:30000/`.

요청 시 KOTS 관리자 암호를 입력한 다음 다음 절차를 수행하여 추가 노드를 생성합니다.

1. 로그인한 후 KOTS 관리 콘솔의 왼쪽 상단에 있는 클러스터 관리 페이지로 이동합니다.
2. [Add node]를 선택합니다.
3. 아래에서 컨트롤러를 선택 취소합니다`Roles`.
4. 아래에서 호출을 선택합니다. `Roles`
5. 제공된 지침에 따라 추가하려는 새 노드에서 명령을 실행합니다.
6. 완료되면 닫기를 선택합니다.
7. 새 노드가 호출 역할과 함께 노드 목록에 나타납니다.
8. KOTS 관리 콘솔의 왼쪽 상단에 있는 애플리케이션 페이지로 이동합니다.
9. 페이지 상단의 탐색 모음에서 구성을 선택합니다.
10. 왼쪽 탐색 패널에서 호출 섹션으로 이동합니다.
11. 호출 노드 사용을 허용하려면 호출 노드 요구하기를 선택합니다.
12. 페이지 하단으로 스크롤하여 구성 저장을 선택합니다.
13. Config가 업데이트되었음을 나타내는 팝업이 나타납니다. 업데이트된 버전으로 이동을 선택합니다.

14. 업데이트된 버전 페이지에 현재 설치된 버전이 표시됩니다. 새 행 항목은 Config Change라는 이름으로 설치된 버전 아래에 나열됩니다. 배포를 선택하여이 새 버전을 배포하고 새 호출 노드를 활성화합니다.

KOTS 관리자 콘솔 구성

KOTS 관리자 콘솔은 처음에 자체 서명된 인증서를 사용하므로 브라우저에서 예외로 허용해야 합니다. 이 예외를 수락하면 KOTS 관리자 콘솔의 구성 마법사에서 환영을 받습니다. 이 마법사는 필요한 경우 사용자 지정 인증서를 추가하는 옵션을 포함하여 KOTS Admin Console의 동작을 구성하기 위한 추가 구성 단계를 안내합니다.

KOTS 관리자 콘솔의 초기 구성이 완료되면 설치 프로세스 중에 생성한 관리자 콘솔 암호를 입력하라는 메시지가 표시됩니다. 처음 로그인할 때 클러스터를 구성해야 합니다.

계속을 선택하여 Wickr용 KOTS 관리자 콘솔로 이동합니다.

단일 노드 임베디드 클러스터의 경우 계속을 선택하여 Wickr용 KOTS 관리자 콘솔로 이동합니다. 다중 노드 설치는 다중 노드 설치를 참조하세요.

KOTS 관리자 콘솔에서 필요에 따라 설치를 구성합니다. 임베디드 클러스터 제품을 사용할 때는 Wickr Enterprise 설치의 적절한 기능을 보장하기 위해 설정해야 하는 몇 가지 주요 구성 설정이 있습니다.

- 호스트 이름 - Wickr 설치와 통신할 때 사용하는 호스트 이름입니다. 이 도메인이 Wickr Enterprise 설치를 가리키도록 적절한 DNS 레코드를 생성해야 합니다.
- 고급 옵션에서 [ ] 수신 컨트롤러 구성 옵션을 선택하여 Kubernetes 수신을 구성하기 위한 구성 블록을 표시합니다. 수신 구성 블록에서 단일 노드 임베디드 클러스터를 선택한 다음 로드밸런서 외부 IP(IPv4 전용)라는 레이블이 지정된 텍스트 상자에 Wickr 서버와 연결된 "퍼블릭" IP를 입력합니다.

  이 IP가 무엇인지 확실하지 않은 경우 Wickr 서버의 명령줄에서 다음 명령을 실행하여이 값을 확인할 수 있습니다. `ip route get 1.1.1.1|awk '{print $7}'`
- 고급 옵션에서 낮은 리소스 모드 활성화 옵션을 선택합니다.
- 호출에서 단일 노드 임베디드 클러스터를 사용하는 경우 호출 노드 요구가 선택 취소되었는지 확인합니다. 그렇지 않으면 초기 설정 중에 호출 노드를 추가한 경우 호출 노드 요구가 선택되어 있는지 확인합니다.
- 파일 공유에 외부 데이터베이스 또는 S3 호환 스토리지를 사용하지 않는 올인원 솔루션을 원하는 경우 다음 설정에 대한 내부 옵션을 선택합니다.
  - Database

- S3 스토리지 위치

  내부 S3 스토리지 위치는 스토리지 용량을 구성하기 위한 추가 옵션을 제공합니다. 축소는 프로비저닝 후 옵션이 아니므로 필요에 따라 작게 시작하고 확장하는 것이 좋습니다.

필요한 모든 기능을 구성했으면 구성 페이지 하단으로 스크롤하여 구성 저장을 선택합니다. 그러면 몇 가지 사전 호스트 검사가 시작됩니다. 사전 검사가 완료되면 배포를 선택하여 Wickr Enterprise 설치를 시작합니다.

이제 Wickr Enterprise 설치 구성을 시작할 준비가 되었습니다. Wickr Enterprise 구성에 대한 자세한 내용은 Wickr Enterprise란 무엇입니까?를 참조하세요.

추가 일반 설치 요구 사항

IP 호스트 이름 설치

설치에 IP 기반 호스트 이름이 필요한 경우 몇 가지 추가 구성 옵션이 있습니다. 이 지침은 IP 기반 호스트 이름에만 적용되며 위에 나열된 기본 설정에 대한 다른 지침을 따르는 것이 좋습니다.

KOTS 관리자 패널에서 다음 단계를 완료합니다.

1. 호스트 이름을 사용할 IP로 설정합니다.
2. 인증서에서 인증서 업로드를 선택합니다. 그런 다음 IP 기반 인증서에 대한 지침에 따라 자체 서명된 인증서를 생성합니다. 자세한 내용은 자체 서명된 인증서 생성을 참조하세요.
3. 인증서용 `.crt` 파일과 프라이빗 키용 `.key` 파일 업로드
4. 인증서 체인의 경우 `.crt` 파일을 다시 업로드합니다.
5. 고정된 인증서 설정 확인란을 선택합니다.
6. 고정된 인증서에 `.crt` 대한를 업로드합니다.
7. 호출에서 서버 퍼블릭 IP 주소 자동 검색 및 트래픽 호출에 호스트 기본 IP 주소 사용 확인란의 선택을 취소합니다.
8. 호출에서 호스트 이름 재정의 텍스트 상자에 호스트 이름의 IP 주소를 입력합니다.
9. 고급 옵션에서 수신 컨트롤러 구성 확인란을 선택합니다. 수신이라는 새 구성 섹션이 아래에 나타납니다.
10. 수신에서 단일 노드 임베디드 클러스터를 선택합니다.
11. 수신에서 Wickr 서버의 '퍼블릭' 인터페이스에 대한 IP를 입력합니다. 호스트 이름으로 사용되는 IP와 다를 수 있습니다. 기본 구성 단계에서이 값에 대한 추가 정보를 참조하세요.

12. 수신에서 와일드카드 호스트 이름 사용을 선택합니다.

SELinux 적용 모드

적용 모드에서 SELinux를 사용해야 하는 경우 임베디드 클러스터를 설치하는 데 사용되는 기본 데이터 디렉터리를 수정합니다. 이 사용 사례에서는 대부분의 SELinux 정책에서 작동하도록 테스트되었으므로를 사용하는 `/opt` 것이 좋습니다.

```
mkdir /opt/wickr
./wickr-enterprise-ha install --license license.yaml --data-dir /opt/wickr --ignore-host-preflights
```

복제된 임베디드 클러스터 기본 설치 사전 검사는 SELinux가 허용 모드인지 확인하고 SELinux가 강제 적용 중이면 실패합니다. 이를 우회하려면 `--ignore-host-preflights` 명령줄 인수를 사용해야 합니다. 명령줄 옵션을 사용하는 경우 아래 프롬프트와 유사합니다. 메시지가 표시되면 예를 입력합니다.

```
#  1 host preflight failed

•  SELinux must be disabled or run in permissive mode. To run SELinux in permissive
 mode, edit /etc/selinux/config, change the line
'SELINUX=enforcing' to 'SELINUX=permissive', save the file, and reboot. You can run
 getenforce to verify the change."

? Are you sure you want to ignore these failures and continue installing? Yes
```

AirGap 설치

Wickr Enterprise의 임베디드 클러스터 설치 옵션은 에어갭 설치를 지원합니다. 라이선스에 대한 추가 구성 및 활성화가 필요합니다. 에어갭 환경에서 Wickr Enterprise 임베디드 클러스터를 사용하는 데 관심이 있는 경우 지원팀에 문의하세요.

에어갭 설치를 수행할 때 다운로드 지침은 표준 설치 방법과 다릅니다. 다음과 유사해야 합니다.

```
curl -f "https://replicated.app/embedded/wickr-enterprise-ha/stable/6.52?airgap=true" -H "Authorization: [redacted]" -o wickr-enterprise-ha-stable.tgz
```

인터넷에 액세스할 수 있는 시스템에 번들을 다운로드한 다음 선호하는 데이터 전송 방법을 사용하여 에어갭 환경으로 전송합니다. 번들이 전송되면 표준 설치 번들과 마찬가지로 추출합니다. 연결된 모든 Wickr Enterprise 애플리케이션 서비스 이미지가 `wickr-enterprise-ha.airgap`포함된 세 번째 파일이 포함됩니다.

```
tar xvf wickr-enterprise-ha-stable.tgz
```

설치하는 동안 추출 후 `--airgap-bundle` 명령줄 인수를 설정해야 합니다. 그렇지 않으면 프로세스가 표준 설치 절차를 따릅니다.

```
./wickr-enterprise-ha install --license license.yaml --airgap-bundle wickr-enterprise-
ha.airgap
```

airGapped 임베디드 클러스터 업데이트

AirGapped Embedded 클러스터를 업데이트하려면 다음 단계를 완료하세요.

1. 복제됨에서 새 임베디드 클러스터 패키지를 다운로드하고 에어갭 환경의 표준 데이터 전송 방법을 사용하여 호스트 시스템으로 전송합니다. 새 번들이 호스트 시스템에 있으면 tarball을 추출합니다.

   ```
   tar xvf wickr-enterprise-ha-stable.tgz
   ```

2. 새 바이너리 및 에어갭 번들을 사용하여 업데이트를 실행합니다.

   ```
   ./wickr-enterprise-ha update --airgap-bundle wickr-enterprise-ha.airgap
   #  Application images are ready!
   #  Finished!
   ```

3. KOTS 관리자 콘솔을 시작하고 KOTS 관리자 콘솔에 액세스하는 표준 방법을 사용하여 제공된 URL에 로그인합니다.

   ```
   ./wickr-enterprise-ha admin-console
   ```

4. KOTS 관리 콘솔에 로그인한 후 버전 아래의 왼쪽에서 사용 가능한 최신 업데이트를 찾은 다음 버전 기록으로 이동 버튼을 누릅니다.

5. 사용 가능한 업데이트에서 새 버전에 대해 배포를 선택합니다. 화면을 살펴봅니다.

    1. 구성 옵션을 변경하고 아래로 스크롤한 후 다음을 선택합니다.
    2. 사전 검사에 실패하지 않았는지 확인하고 다음: 확인 및 배포를 선택합니다.
    3. 배포(Deploy)를 선택합니다.

Wickr Enterprise 임베디드 클러스터에 대한 추가 참고 사항

- NAMESPACE: 대부분의 Wickr Enterprise 설치와 달리 임베디드 클러스터 설치는 Wickr 자산을 Wickr가 아닌 kubernetes의 kotsadm 네임스페이스에 설치합니다. kubectl, helm 또는 기타 유틸리티`-n wickr`에서 `-n kotsadm` 대신 사용하도록 저장한 스크립트 또는 명령을 수정합니다.
- Kubernetes 클러스터와 상호 작용: 호스트 시스템에서 `./wickr-enterprise-ha` 바이너리를 사용하여를 실행하여 Kubernetes 설치와 상호 작용하도록 설정된 적절한 변수가 있는 쉘을 생성합니다./wickr-enterprise-ha shell. 그러면 쉘의 PATH 내에 kubectl 유틸리티가 제공되고 적절한 kube 구성이 로컬 설치로 설정됩니다.

Wickr 임베디드 클러스터 설치 문제 해결

이러한 문제 해결 단계의 모든 인스턴스는 Wickr Embedded Cluster 설치를 실행하는 인스턴스에 대한 셸 액세스 권한이 있고 `./wickr-enterprise-ha shell` 명령을 실행하여 Kubernetes 설치와 직접 상호 작용할 수 있다고 가정합니다.

일반 문제

클러스터 관리 화면에서 노드 추가 버튼 누락

에어갭 설치

에어갭을 설치하는 경우 Wickr Support에 문의하여이 동작을 수정하는 데 도움을 받으세요.

표준 설치

라이선스에 임베디드 클러스터 다중 노드 권한이 포함된 경우 라이선스 동기화를 수행하여 최신 버전을 가져옵니다. 확실하지 않거나이 권한이 없는 경우 Wickr Support에 문의하십시오.

라이선스 동기화를 수행하려면 다음 단계를 완료합니다.

1. KOTS 제어판으로 이동합니다.

2. 대시보드 페이지의 오른쪽 상단 영역에서 라이선스 섹션을 찾습니다.
3. 이 섹션의 오른쪽 상단에는 동기화 라이선스 하이퍼링크가 표시됩니다. 하이퍼링크를 선택합니다.
4. 라이선스가 동기화되면 UI가 업데이트되고 몇 초 전에 마지막으로 동기화된가 나타납니다.
5. KOTS 대시보드 페이지의 버전 섹션에서 재배포를 선택합니다.
6. 재배포가 완료되면 클러스터 관리로 돌아가 노드를 추가할 수 있습니다.

업그레이드 문제

클러스터 업그레이드 시 업그레이드 중단

업그레이드가 클러스터 업그레이드에 멈춘 경우 일부 포드가 적절하게 종료되지 않을 수 있습니다. 인스턴스에 로그온하고 `./wickr-enteprise-ha shell` 명령을 사용하여 kubernetes 설치를 관리하기 위한 쉘 환경에 들어갑니다.

1. 아직 실행 중인 포드를 식별합니다.

   ```
   kubectl -n kotsadm get pods | grep Running
   ```

2. `kubectl -n kotsadm delete pod name-of-running-pod`

   > **Note**
   >
   > 실행 중인 포드 중 하나가 `embedded-cluster-upgrade-XXXXXXXXXXXXXX-xxxxx` `kotsadm-xxxxxxx` 또는 유사한 경우 업그레이드를 수행하는 데 필요하므로 삭제하지 마십시오.

3. 실행 중인 포드가 남아 있지 않은지 확인합니다.

   ```
   kubectl -n kotsadm get pods | grep Running
   ```

이 절차에서는 클러스터 업그레이드가 Wickr 업그레이드로 진행되도록 허용해야 합니다.

클러스터 업그레이드 중에 애플리케이션이 업데이트되지 않아 새 버전을 배포할 수 없음

업그레이드 후에도 애플리케이션이 이전 버전을 유지하는 경우 새 버전이 일관되지 않은 상태일 수 있습니다.

Kubernetes 설치 레코드를 확인합니다.

1. 설치 프로그램에서 Kubernetes 쉘을 엽니다.

   ```
   ./wickr-enterprise-ha shell
   ```

2. 다음 kubectl 명령을 실행합니다.

   ```
   kubectl get installations
   ```

3. 출력은 다음과 같습니다.

   ```
   [root@ip-172-31-6-72 ~]# kubectl get installations
   NAME              STATE      INSTALLERVERSION   CREATEDAT              AGE
   20251113170603    Obsolete   2.1.3+k8s-1.30     2025-11-13T17:06:05Z   22h
   20251113180133    Failed     2.6.0+k8s-1.31     2025-11-13T18:01:37Z   21h
   ```

4. 실패한 설치를 삭제합니다.

   ```
   kubectl delete installation 20251113180133
   ```

5. KOTS 관리자 패널을 통해 업그레이드를 다시 실행해 봅니다.

로그 줄을 사용한 RabbitMQ 포드 실패 **Error while waiting for Mnesia tables: {timeout_waiting_for_tables}**

RabbitMQ 보안 암호와 스토리지가 동기화되지 않았습니다. 이는 일반적으로 여러 RabbitMQ 인스턴스가 실행되어 리더 선택 또는 쿼럼 오류가 발생할 때 발생합니다. 이 문제를 해결하려면 RabbitMQ 서비스와 해당 스토리지 볼륨을 삭제한 다음 재배포합니다.

장애가 발생한 RabbitMQ를 삭제하려면 다음 단계를 완료합니다.

1. RabbitMQ 상태 저장 세트를 삭제합니다.

   ```
   kubectl -n kotsadm delete statefulset rabbitmq –cascade=orphan
   ```

2. 나머지 RabbitMQ 포드를 삭제합니다. 실행 중인 RabbitMQ-X 포드가 여러 개 있는 경우이 명령을 여러 번 실행하여 추가 포드 이름에 맞게 RabbitMQ-X 값을 업데이트합니다.

   ```
   kubectl -n kotsadm delete pod rabbitmq-0
   ```

3. 해당 PVCs. 실행 중인 포드가 여러 개 있는 경우이 명령을 여러 번 실행하여 data-RabbitMQ-X를 적절한 포드에 맞게 업데이트합니다.

   ```
   kubectl -n kotsadm delete pvc data-rabbitmq-0
   ```

4. 남은 포드가 있는지 확인합니다. 성공하면 아무것도 출력하지 않습니다.

```
kubectl -n kotsadm get pods|grep -i rabbitmq
```

5. 나머지 PVCs가 있는지 확인합니다. 성공하면 아무것도 출력하지 않습니다.

```
kubectl -n kotsadm get pvc|grep -i rabbitmq
```

6. KOTS 관리 패널을 통해 재배포합니다.

자세한 문제 해결 정보는 문제 해결을 참조하세요.

문서 이력

다음 표에서는 Wickr Enterprise 자동 설치 안내서의 설명서 릴리스를 설명합니다.

변경 사항	설명	날짜
보안 설정	보안 설정이 추가되었습니다. 자세한 내용은 보안 설정을 참조하세요.	2025년 8월 26일
다중 노드 설치	다중 노드 설치가 추가되었습니다. 자세한 내용은 다중 노드 설치를 참조하세요.	2025년 8월 26일
수신 설정 호출	호출 수신 설정이 추가되었습니다. 자세한 내용은 수신 설정 호출을 참조하세요.	2025년 8월 26일
자동 배포 옵션	자동 배포 옵션이 추가되었습니다. 자세한 내용은 Wickr Enterprise 설치를 참조하세요.	2024년 2월 23일
허용 목록에 대한 포트	포트 TCP/8443이 허용 목록에 추가되었습니다. 자세한 내용은 요구 사항을 참조하세요.	2024년 2월 12일
허용 목록에 리소스 및 포트 폐기	리소스를 삭제하는 방법에 대한 지침이 추가되었습니다. 자세한 내용은 리소스 폐기를 참조하세요. 또한 허용 목록에 대한 포트가 추가되었습니다. 자세한 내용은 요구 사항을 참조하세요.	2023년 8월 17일
최초 릴리스	Wickr Enterprise 자동 설치 안내서의 최초 릴리스	2023년 8월 4일

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