aws

自動インストールガイド

Wickr エンタープライズ

Wickr エンタープライズ: 自動インストールガイド

Table of Contents

Wickr エンタープライズとは何ですか？

Wickr エンタープライズ は、組織や政府機関が 1 対 1 およびグループのメッセージング、音声およびビデオ通話、ファイル共有、画面 共有を通じて安全に通信できるようにする、エンドツーエンドの暗号化されたセルフホスト型サービスです。顧客は Wickr エンタープライズ を利用することで、コンシューマーグレードのメッセージングアプリに関連するデータ保持義務を克服し、安全にコラボレーションを促進することができます。高度なセキュリティと管理制御により、組織は法的要件や規制要件を満たし、データセキュリティの課題に対応するカスタムソリューションを構築できます。

情報は、保存や監査の目的で、カスタマーが管理するプライベートなデータストアに記録できます。顧客は、権限の設定、エフェメラルメッセージングオプションの設定、セキュリティグループの定義など、データを包括的に管理できます。また、管理者は Wickr ボットを使用してワークフローを安全に自動化できます。Wickr エンタープライズ は、Active Directory や OpenID Connect (OIDC) によるシングルサインオン (SSO) などの追加サービスと統合されています。Wickr エンタープライズ の設定を開始するには、「Wickr エンタープライズ の概要」を参照してください。

> **Note**
>
> Wickr エンタープライズ デプロイパッケージをまだお持ちでない場合は、「ビジネスに関するお問い合わせ先」を参照してください。

Wickrエンタープライズの開始方法

トピック

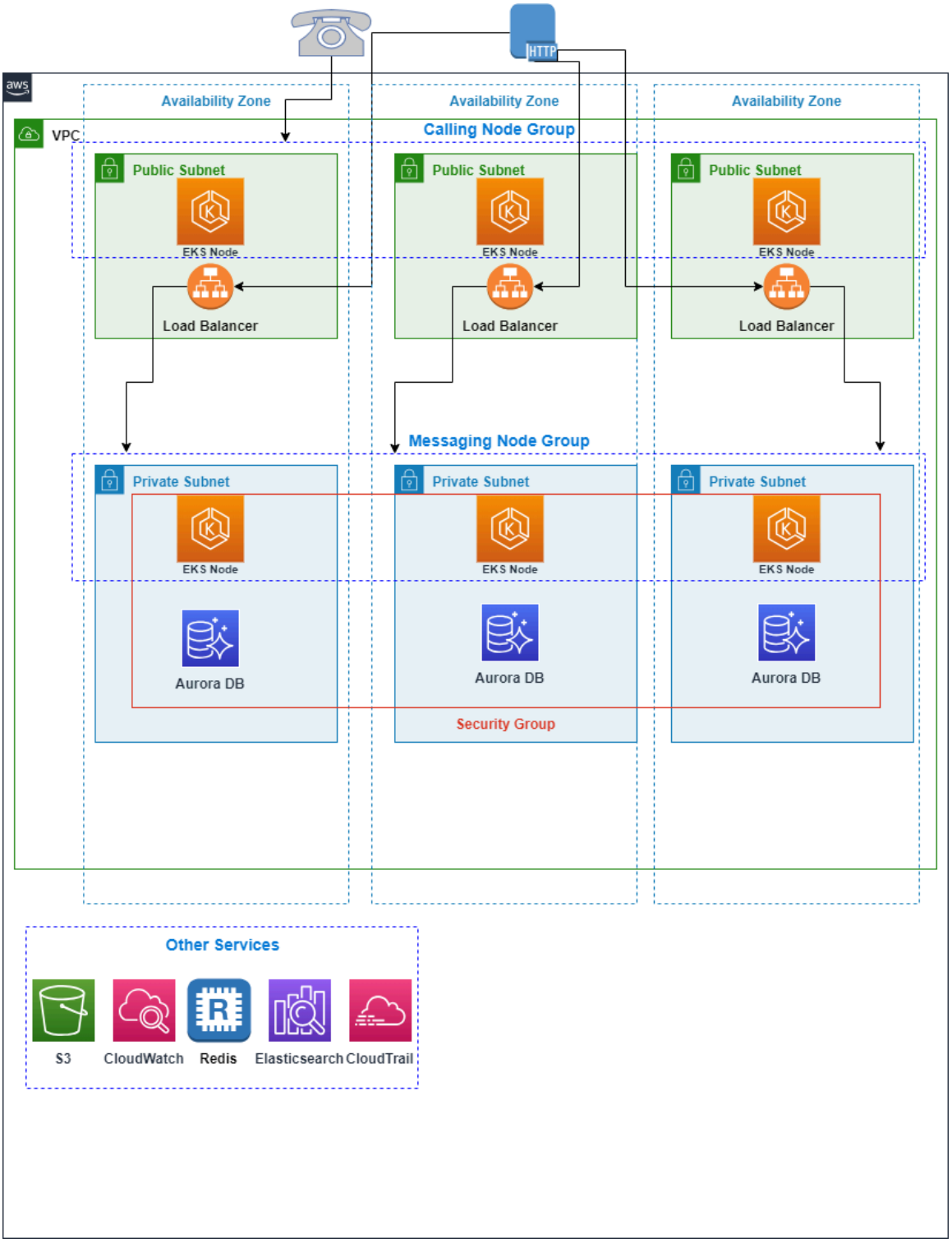
要件

開始する前に、以下の要件が満たされていることを確認します:

- Node.js 16+ をダウンロードします
- AWS CLI アカウントの認証情報で設定されている。

  これらは、`~/.aws/config` にある設定ファイルから、または `AWS_` 環境変数を使用して取得されます。
- kubectl をインストールします。詳細については、Amazon EKS ユーザーガイドの「kubectl のインストール」を参照してください。
- KOTS CLI をインストールします。詳しくは「kots CLIのインストール」を参照してください。
- 許可リストへのポート: HTTPS および TCP 呼び出しトラフィックの場合は 443/TCP、UDP 呼び出しトラフィックの場合は 16384-19999/UDP、TCP/8443

アーキテクチャ



依存関係をインストールします

以下のコマンドを使用して、すべての依存関係をデフォルトのパッケージに追加できます。

```
npm install
```

構成する

AWS Cloud Development Kit (AWS CDK) は、コンテキスト値を使用してアプリケーションの設定を制御します。Wickr Enterprise は CDK コンテキスト値を使用して、インストールされている Wickr エンタープライズのドメイン名や RDS バックアップを保持する日数などの設定を制御します。詳しくは、「AWS Cloud Development Kit (AWS CDK) デベロッパーガイド」の「ランタイムコンテキスト」を参照してください。

コンテキスト値を設定するには複数の方法がありますが、特定のユースケースに合わせて `cdk.context.json` で値を編集することをお勧めします。`wickr/` で始まるコンテキスト値のみが Wickr Enterprise デプロイに関連し、残りは CDK 固有のコンテキスト値です。次回 CDK を使用して更新を行うときに同じ設定を維持するには、このファイルを保存します。

少なくとも、`wickr/licensePath`、、`wickr/domainName`および `wickr/acm:certificateArn`または `wickr/route53:hostedZoneId`と を設定する必要がありま す`wickr/route53:hostedZoneName`。

パブリックホストゾーンを使用する場合

に Route 53 パブリックホストゾーンがある場合は AWS アカウント、次の設定を使用して CDK コンテキストを設定することをお勧めします。

- `wickr/domainName`: この Wickr Enterprise デプロイに使用するドメイン名。Route 53 パブリックホストゾーンを使用する場合、このドメイン名の DNS レコードと ACM 証明書は自動的に作成されます。
- `wickr/route53:hostedZoneName`: DNS レコードを作成する Route 53 ホストゾーン の名前。
- `wickr/route53:hostedZoneId`: DNS レコードを作成する Route 53 ホストゾーン ID。

この方法では、Wickr Enterprise デプロイの前にあるロードバランサーにドメイン名を指定する DNS レコードとともに、ユーザーに代わって ACM 証明書を作成します。

パブリックホストゾーンがない場合

アカウントに Route 53 パブリックホストゾーンがない場合は、ACM 証明書を手動で作成し、`wickr/acm:certificateArn` コンテキスト値を使用して CDK にインポートする必要があります。

- `wickr/domainName`: この Wickr Enterprise デプロイに使用するドメイン名。Route 53 パブリックホストゾーンを使用する場合、このドメイン名の DNS レコードと ACM 証明書は自動的に作成されます。
- `wickr/acm:certificateArn`: ロードバランサーで使用する ACM 証明書の ARN。アカウントで Route 53 パブリックホストゾーンを利用できない場合は、この値を指定する必要があります。

への証明書のインポート

次のコマンドを使用して、外部から取得した証明書をインポートできます。

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

出力は証明書 ARN になり、`wickr/acm:certificateArn` コンテキスト設定の値として使用する必要があります。アップロードした証明書が `wickr/domainName` で有効であることが重要です。そうしないと、HTTPS 接続を検証できなくなります。詳細については、AWS Certificate Manager ユーザーガイドの 証明書のインポート を参照してください。

DNS レコードの作成

利用可能なパブリックホストゾーンがないため、デプロイが完了した後に、Wickr Enterprise デプロイの前面にあるロードバランサーを指すように DNS レコードを手動で作成する必要があります。

既存の VPC へのデプロイ

既存の VPC を使用する必要がある場合は、VPC を使用できます。ただし、VPC は EKS に必要な仕様を満たすように設定する必要があります。詳細については、「Amazon EKS ユーザーガイド」の「VPC とサブネットの Amazon EKS ネットワーク要件を表示」を参照してください。また、使用する VPC がこれらの要件を満たしていることを確認します。

さらに、次のサービスの VPC エンドポイントがあることを確認することを強くお勧めします。

- クラウドウォッチ
- CLOUDWATCH_LOGS
- EC2
- EC2_MESSAGES

- ECR
- ECR_DOCKER
- ELASTIC_LOAD_BALANCING
- KMS
- SECRETS_MANAGER
- SSM
- SSM_MESSAGES

リソースを既存の VPC にデプロイするには、以下のコンテキスト値を設定します。

- `wickr/vpc:id` - リソースをデプロイする VPC ID (例: `vpc-412beef`)。
- `wickr/vpc:cidr` - VPC の IPv4 CIDR (例: `172.16.0.0/16`)。
- `wickr/vpc:publicSubnetIds` - VPC 内のパブリックサブネットのカンマ区切りリスト。Application Load Balancer と呼び出し側の EKS ワーカーノードは、これらのサブネットにデプロイされます (例: `subnet-6ce9941,subnet-1785141,subnet-2e7dc10`)。
- `wickr/vpc:privateSubnetIds` - VPC 内のプライベートサブネットのカンマ区切りリスト。EKS ワーカーノードと踏み台サーバーは、これらのサブネットにデプロイされます (例: `subnet-f448ea8,subnet-3eb0da4,subnet-ad800b5`)。
- `wickr/vpc:isolatedSubnetIds` - VPC 内の隔離されたサブネットのカンマ区切りリスト。RDS データベースはこれらのサブネットにデプロイされます (例: `subnet-d1273a2,subnet-33504ae,subnet-0bc83ac`)。
- `wickr/vpc:availabilityZones` - VPC 内のサブネットのアベイラビリティーゾーンのカンマ区切りリスト (例: `us-east-1a,us-east-1b,us-east-1c`)。

インターフェイス VPC エンドポイントの詳細については、「インターフェイス VPC エンドポイントを使用して AWS サービスにアクセスする」を参照してください。

その他の設定

詳細については、「Context values」を参照してください。

ブートストラップ

この特定の AWS アカウント およびリージョンで CDK を初めて使用する場合は、まずアカウントをブートストラップして CDK の使用を開始する必要があります。

```
npx cdk bootstrap
```

デプロイ

この処理には約 45 分かかります。

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

これが完了すると、インフラストラクチャーが作成され、Wickrエンタープライズのインストールを開始できます。

DNS レコードの作成

CDK の設定時にパブリックホストゾーンを使用する場合、この手順は不要です。

デプロイプロセスの出力には、ロードバランサーの DNS 名の値 `WickrAlb.AlbDnsName` が含まれます。出力は次のようになります。

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

この場合、ロール名は `Wickr-Alb-1Q5IBPJR4ZVZR-409483305.us-west-2.elb.amazonaws.com` です。この値は、ドメイン名の CNAME または A/AAAA (ALIAS) レコードを作成するときに使用する必要があります。

デプロイからの出力がない場合は、次のコマンドを実行してロードバランサーの DNS 名を表示します。

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

KOTS 設定の生成

> **Warning**
>
> このファイルには、インストールに関する機密情報が含まれています。公開で共有したり、保存したりしないでください。

Wickr Enterprise インストーラーを正常にインストールするには、インフラストラクチャーに関するいくつかの設定値が必要です。ヘルパースクリプトを使用して設定値を生成できます。

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

最初のステップで外部証明書を ACM にインポートした場合は、このスクリプトに `--ca-file` フラグを渡します。次に例を示します。

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

スタックが存在しないというエラーが表示された場合は、選択したリージョンに `AWS_REGION` 環境変数 (`export AWS_REGION=us-west-2`) を設定して、もう一度試してください。または、コンテキスト値 を設定した場合は`wickr/stackSuffix`、 `--stack-suffix`フラグでサフィックスを渡します。

Kubernetes クラスターに接続する

Amazon EKS API には、デプロイの一部として作成された拠点ホストからのみアクセスできます。そのため、すべての kubectl コマンドは踏み台ホスト自体で実行するか、踏み台ホスト経由でプロキシする必要があります。

踏み台経由のプロキシ接続

クラスターに初めて接続するときは、aws eks update-kubeconfig コマンドを使用してローカルの kubeconfig ファイルを更新し、設定で proxy-url を設定する必要があります。その後、クラスターに接続するたびに踏み台ホストで SSM セッションを開始し、API アクセスのためにプロキシにポート転送します。

1 回限りのセットアップ

WickrEks CloudFormation スタックには、WickrEnterpriseConfigCommand で始まる名前の出力値があります。値には、クラスターの kubectl 設定を生成するのに必要なコマンドがすべて含まれています。この出力は以下のコマンドで見ることができます。

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

これにより、aws eks update-kubeconfig で始まるコマンドが出力されるはずです。このコマンドを実行する。

次に、踏み台ホスト経由でリクエストをプロキシするように Kubernetes 設定を変更する必要があります。これは、次のコマンドを使用して実行できます。

```
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
```

正しく動作していれば、'Property "clusters.arn:aws:eks:us-west-2:012345678912:cluster/

WickrEnterprise5B8BF472-1234a41c4ec48b7b615c6789d93dcce.proxy-url" set.' のような出力が表示されます。

踏み台へのポート転送

Amazon EKS クラスターに接続するには、SSM セッションを開始して、要塞ホストで実行されているプロキシに転送リクエストをポート転送する必要があります。これを行うコマンドは、BastionSSMProxyEKSCommand スタックの出力 WickrEks として提供されます。以下のコマンドを実行して、出力値を表示します。

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

出力されるコマンドは aws ssm start-session で始まります。このコマンドを実行して、ポート 8888 で実行されているローカルプロキシを起動し、これを使用して Amazon EKS クラスターに接続できます。ポートフォワードが正常に機能していれば、出力には「接続待ち...」と表示されるはずです。Amazon EKS クラスターにアクセスする必要がある間は、このプロセスを実行し続けてください。

すべてが正しく設定されている場合、別のターミナルkubectl get nodesで を実行して、Amazon EKS クラスター内のワーカーノードを一覧表示できます。

```
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エンタープライズのインストール

Kubernetes クラスターへの接続が確立されたら、kubectl kots プラグインを使用して Wickrエンタープライズのインストールを開始できます。KOTS ライセンスファイル (Wickr が提供する .yaml ファイル) と設定値ファイルが必要です。これらは KOTS 設定生成 セクションの wickr-config.json ファイルに保存されています。KOTS 設定生成について詳しくは、「KOTS 設定生成」を参照してください。

Wickr Enterprise の手動インストール

次のコマンドを実行すると、Wickrエンタープライズのインストールが開始されます。

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

KOTS Admin Consoleのパスワードを入力するプロンプトが表示されます。このパスワードは、今後Wickrエンタープライズのインストール設定をアップグレードまたは変更する際に必要になるため、保存してください。

インストールが完了すると、kubectl kots が KOTS 管理コンソールへのアクセスを提供するローカルポート (通常は http://localhost:8080) を開きます。このサイトで Wickrエンタープライズのインストール状況を変更または監視したり、ブラウザでインストール用に設定したドメイン名にアクセスして Wickr のセットアップを開始したりできます。

Lambda を使用した Wickr Enterprise のインストール

CDK のデプロイ中に、ユーザーに代わって Wickr Enterprise のインストールを自動的に完了するために Lambda が作成され、呼び出されます。手動で呼び出すには、 AWS コンソールを開き、Lambda WickrLambda-func* 関数を見つけます。テストタブで、 を選択します。test入力は無関係です。

インストール後

Wickrエンタープライズのインストールを管理するには、KOTS 管理コンソールと Wickr 管理コンソールの 2 つの Web コンソールを使用できます。

> **Note**
>
> 組織のバックアップおよびロギングポリシー（Amazon S3 設定、Elastic Load Balancing アクセスログ、 Amazon Virtual Private Cloud フローログ）を反映するために必要な変更を行います。

KOTS 管理コンソール

このインターフェースは Wickrエンタープライズのデプロイされたバージョンを管理するために使用されます。インストールのステータスを確認したり、設定を変更したり、アップグレードを実行したりできます。KOTS 管理コンソールには Kubernetes ポート転送からのみアクセスできます。ポート転送は以下のコマンドで開くことができます。

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

> **Note**
>
> まず、踏み台へのポート転送セクションで説明されているように、踏み台接続を設定する必要があります。踏み台へのポート転送について詳しくは、「踏み台経由のプロキシ接続」を参照してください。

ポート転送が正常に設定されると、前のコマンドは以下を出力します。

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

提供された URL を使用して KOTS 管理コンソールにアクセスします。ログイン用のパスワードは、インストール中に `kubectl kots install` を実行したときに選択したパスワードです。パスワードをリセットする必要がある場合は、「KOTS 管理コンソールパスワードのリセット」を参照してください。

Wickr 管理コンソール

このインターフェースは、Wickrエンタープライズのインストール環境を設定してネットワーク、ユーザ、フェデレーションを設定するために使用されます。ロードバランサーを指すように設定した DNS 名で HTTPS 経由でアクセスできます。DNS がパブリックホストゾーンで自動的に設定された場合、ドメイン名は `wickr/domainName` コンテキスト値の値です。

デフォルトのユーザー名は `admin`、パスワードは `Password123` です。このパスワードは、初回ログイン時に変更する必要があります。

コンテキスト値

コンテキスト値は、アプリ、スタック、またはコンストラクトに関連付けることのできるキーと値のペアです。これらは、ファイル (通常はプロジェクトディレクトリ内の `cdk.json` または `cdk.context.json`) またはコマンドラインからアプリに提供できます。CDK はコンテキスト値を使用してアプリケーションの設定を制御します。Wickr Enterprise は CDK コンテキスト値を使用して、インストールされている Wickrエンタープライズのドメイン名や RDS バックアップを保持する日数などの設定を制御します。

コンテキスト値を設定するには複数の方法がありますが、特定のユースケースに合わせて `cdk.context.json` で値を編集することをお勧めします。Wickr Enterprise `wickr/` のデプロイに関連するのは、 で始まるコンテキスト値だけです。

名前	説明	デフォルト
`wickr/licensePath`	KOTS ライセンスへのパス (Wickr が提供する `.yaml` ファイル)。	null
`wickr/domainName`	この Wickr Enterprise デプロイに使用するドメイン名。Route 53 パブリックホストゾーンを使用する場合、このドメイン名の DNS レコードと ACM 証明書は自動的に作成されます。	null
`wickr/route53:hostedZoneId`	DNS レコードを作成する Route 53 ホストゾーン ID。	null
`wickr/route53:hostedZoneName`	DNS レコードを作成する Route 53 ホストゾーン の名前。	null
`wickr/acm:certificateArn`	ロードバランサーで使用する ACM 証明書の ARN。アカウントで Route 53 パブリックホストゾーンを利用できない場	null

名前	説明	デフォルト
	合は、この値を指定する必要があります。	
wickr/caPath	証明書パス。自己署名証明書を使用する場合にのみ必要です。	null
wickr/vpc:id	リソースをデプロイする VPC の ID。既存の VPC にデプロイする場合にのみ必要です。設定を解除すると、新しい VPC が作成されます。	null
wickr/vpc:cidr	作成されたVPCに関連付ける IPv4 CIDR。既存の VPC にデプロイする場合は、これを既存の VPC の CIDR に設定します。	172.16.0.0/16
wickr/vpc:availabilityZones	アベイラビリティーゾーンのカンマ区切りリスト。既存の VPC にデプロイする場合にのみ必要です。	null
wickr/vpc:publicSubnetIds	パブリックサブネット ID のカンマ区切りリスト。既存の VPC にデプロイする場合にのみ必要です。	null
wickr/vpc:privateSubnetIds	プライベートサブネット ID のカンマ区切りリスト。既存の VPC にデプロイする場合にのみ必要です。	null

名前	説明	デフォルト
wickr/vpc:isolatedSubnetIds	RDS データベースの隔離されたサブネット ID のカンマ区切りリスト。既存の VPC にデプロイする場合にのみ必要です。	null
wickr/rds:deletionProtection	RDS インスタンスの削除保護を有効にします。	true
wickr/rds:removalPolicy	RDS インスタンスである「スナップショット」、「破棄」、または「保持」の削除ポリシー。	スナップショット
wickr/rds:readerCount	RDS クラスターで作成するリーダーインスタンスの数。	1
wickr/rds:instanceType	RDS インスタンスに使用するインスタンスタイプ。	r6g.xlarge
wickr/rds:backupRetentionDays	バックアップを保持する日数。	7
wickr/eks:namespace	EKS の Wickr サービスのデフォルト名前空間。	wickr
wickr/eks:defaultCapacity	メッセージングインフラストラクチャの EKS ワーカーノードの数。	3
wickr/eks:defaultCapacityCalling	通話インフラストラクチャの EKS ワーカーノードの数。	2
wickr/eks:instanceTypes	メッセージング EKS ワーカーノードに使用するインスタンスタイプのコンマ区切りリスト。	m5.xlarge

名前	説明	デフォルト
wickr/eks:instanceTypesCalling	通話 EKS ワーカーノードに使用するインスタンスタイプのコンマ区切りリスト。	c5n.large
wickr/eks:enableAutoscaler	EKS の Cluster Autoscaler 機能を有効にするかどうかを切り替えます。	true
wickr/s3:expireAfterDays	ファイルのアップロードが S3 バケットから削除されるまでの日数。	1095
wickr/eks:clusterVersion	クラスターバージョン (Kubernetes バージョン、kubectlLayer バージョン、albController バージョン、nodeGroupRelease バージョンなどを含む)。	1.27
wickr/stackSuffix	CloudFormation スタック名に適用するサフィックス。	"
wickr/autoDeployWickr	Lambda を使用して Wickr アプリケーションを自動デプロイします。	true

リソースの破壊

この AWS CDK アプリケーションによって作成されたすべてを削除するには、他のすべてのWickrRdsスタックの前にスタックを削除する必要があります。

Amazon RDS リソースを適切に削除するには、削除保護を無効にし、削除ポリシーを snapshot または destroy に設定する必要があります。これらが現在の設定でない場合は、 AWS CDK コンテキストで wickr/rds:deletionProtection および wickr/rds:removalPolicy の値を変更し、npx cdk deploy -e WickrRds を実行して Amazon RDS スタックを再デプロイします。

削除保護と削除ポリシーが適切に設定されたら、WickrRds スタックに対して cdk destroy を実行します。

```
npx cdk destroy WickrRds
```

スタックの破棄が終了したら、残りの CloudFormation WickrRds スタックを以下のコマンドで破棄できます。

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

トラブルシューティング

Wickr ネームスペースの削除

wickr ネームスペースを削除して最初からやり直す必要がある場合は、まず CDK がそのネームスペース内で作成したサービスアカウントをバックアップすることが重要です。これらのサービスアカウントにより、Wickr サービスは IAM ロールを介して AWS APIs と通信できます。これらがないと、Amazon Simple Storage Service (Amazon S3) によるファイルのアップロードなどのタスクは機能しなくなります。

以下のコマンドを使用してサービスアカウントをバックアップし、wickrネームスペースと適切なサービスアカウントを削除して再作成します。

```
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 管理コンソールのパスワードをリセットする

KOTS 管理コンソールのパスワードは、以下のコマンドでリセットできます。

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

このパスワードを変更すると、wickr/kotsSecrets Manager シークレットも更新できますが、通常、自動化によって再び使用されることはありません。

踏み台を使用した EKS クラスターへの接続に関する問題

踏み台を介した EKS クラスターへの接続が遅いか、時折タイムアウトする場合は、kubectlコマンドの実行時に次のエラーが表示されることがあります。

net/http: 接続の待機中にリクエストがキャンセルされました (Client.Timeout exceeded while awaiting headers)

この問題は、多くの場合、SSM 経由で踏み台ホストにログインし (WickrEks スタックBastionSSMCommandの を参照)、tinyproxyサービスを再起動することで解決できます。

```
sudo systemctl restart tinyproxy
```

カスタムインストール

カスタムインストールセクションでは、Wickr Enterprise をインストールする方法について説明します。

トピック

要件

Wickr Enterprise のインストールを開始する前に、以下の要件を満たしていることを確認してください。

ハードウェア要件

Wickr Enterprise を使用するには、Kubernetes クラスターが必要です。Low Resource Mode が有効になっている 1 つのノードで操作することはできますが、一般的な本番稼働用にはお勧めしませ

ん。本番稼働用デプロイでは、最低 3 つのメッセージングワーカーノードと最低 2 つの呼び出しワーカーノードをお勧めします。

ワーカーノードには、次の最小仕様が必要です。

- 2～4 個の CPU コア
- 8 GB の Ram
- 200 GB のディスク容量

最小ハードウェア要件

低リソースモードで実行されている単一のワーカーノードクラスターには、最低 3000m の CPU と 5846Mi Ram が必要です。これには kube-system ポッドは含まれません。

ポッド別のリソース要件

ポッド名	所有者	CPU	メモリ
admin-api	Wickr	100m	256Mi
ディレクトリ	Wickr	100m	128Mi
有効期限	Wickr	100m	128Mi
ファイルプロキシ	Wickr	100m	256Mi
OIDC	Wickr	100m	128Mi
opensearch	Wickr	500 m	100Mi
オルビル	Wickr	50 メートル	128Mi
orville-redis	Wickr	50 メートル	128Mi
プッシュデバイス	Wickr	100m	128Mi
rabbitmq	Wickr	50 メートル	256Mi
対応	Wickr	100m	64Mi
受信	Wickr	250 メートル	128Mi

ポッド名	所有者	CPU	メモリ
redis	Wickr	50 メートル	128Mi
server-api	Wickr	250 メートル	256Mi
スイッチボード	Wickr	250 メートル	512Mi
コツサドム	KOTS	50 メートル	50Mi
コッツアドムミニオ	KOTS	100m	512Mi
kotsadm-rqlite	KOTS	200m	1Gi
minio-operator	内部 S3	200m	256Mi
ミニオテナント	内部 S3	100m	256Mi
mysql-primary	内部 MySQL	100m	512Mi
mysql-secondary	内部 MySQL	100m	512Mi

ストレージ要件

Wickr Enterprise では、永続ボリュームクレームを作成するときに使用するデフォルトの StorageClass が必要です。エアギャップ環境にデプロイする場合やオンプレミスにデプロイする場合は、クラスター用に設定する必要がある場合があります。使用可能なオプションの 1 つは Longhorn です。推奨されるディスク容量の要件は、内部 S3 オプションと内部 Mysql オプションの使用、およびファイルのアップロードに使用できる容量によって異なります。

- 内部イメージキャッシュ: ~60 Gi
- RabbitMQ: 24 Gi デフォルト/8 Gi 低リソースモード
- Redis: 24 Gi デフォルト/8 Gi 低リソースモード
- OpenSearch: 24 Gi のデフォルト/8 Gi の低リソースモード
- 内部 Mysql: 80 Gi デフォルト/20Gi 低リソースモード
- 内部 S3: 160 Gi デフォルト/2Gi 低リソースモード
- KOTS ミニオ: 4 Gi
- KOTS Rqlite: 1 Gi

最小ストレージサイズ

- 内部 S3 と内部 Mysql を使用した 377 Gi のデフォルト
- 低リソースモードで 111 Gi

Kubernetes バージョンの要件

Wickr Enterprise はレプリケートされた KOTS に依存しています。商用ソフトウェアディストリビューションプラットフォームである Replicated は、現在サポートされている Kubernetes のバージョンのリストを提供します。詳細については、「Kubernetes バージョンの互換性」を参照してください。

ソフトウェア要件

Wickr Enterprise を使用するには、Kubernetes クラスターと KOTS が必要です。サポートされている OS および Kubernetes のバージョンについては、KOTS ドキュメントを参照してください。詳細については、「最小システム要件」を参照してください。

開発者ホストシステム

オペレーティングシステム — このドキュメントのコマンドは、WSL (Linux 用 Windows サブシステム) がインストールされた Linux、MacOS、または Windows で動作するように設計されています。

内部ステートフルサービス

Wickr Enterprise は、MySQL データベースと S3 互換ストレージの両方に内部サービスを提供できますが、一般的な本番稼働用には、Kubernetes クラスターの外部でこれらのサービスを提供することをお勧めします。

- MySQL 5.7 データベース
  - Amazon RDS MySQL 5.7 または MySQL 5.7 データベース (外部)
  - Mysql Bitnami Helm チャート (内部)
  - ファイルストレージ
    - Amazon S3 または S3 互換ストレージプロバイダー (外部)
    - Minio Operator Helm チャート (内部)

ネットワークの要件

Wickr Enterprise には、FQDN、SSL 証明書、および特定のオープン TCP および UDP ポートが必要です。

- FQDN: Wickr Enterprise デプロイで使用されるドメインまたはサブドメイン。
- SSL 証明書: パブリック CA または自己署名証明書キーペアによって署名された SSL 証明書キーペア。証明書は、共通名および SAN DNS エントリとして FQDN をリストする必要があります。証明書は serverAuth extendedKeyUsage 拡張機能も有効にする必要があります。
- オンラインインストールには、レプリケートされたリソースとサードパーティーのリソースへの出力アクセスが必要です。レプリケートされた は、IP アドレスのリストを保持します。詳細については、「レプリケートされた IP アドレス」を参照してください。レプリケートされた は、必要なサードパーティーリソースのリストも保持します。詳細については、「オンラインインストールのファイアウォールオープニング」を参照してください。
- エアギャップインストールでは、プライベートコンテナレジストリにアクセスする必要があります。

メッセージングノード

メッセージングノードはパブリック IPV4 アドレスを必要としないため、プライベートサブネットに配置する必要があります。メッセージトラフィックは LoadBalancer または Ingress を介してクラスターに入ります。

ノードの呼び出し

ノードを呼び出すには、パブリック IPV4 アドレスが必要なため、パブリックサブネットに存在する必要があります。デフォルトでは、通話メディアは UDP 経由で転送されます。TCP 呼び出しが有効になっている場合、TCP Proxy は TCP 443 で接続を受け入れ、Orville サービスにプロキシします。

- TCP : 443 TCP プロキシの呼び出し
- UDP : 16384-16484 オーディオ/ビデオストリーム

インストールと設定のアクセス

インストールと設定のための KOTS 管理コンソールへのアクセスは、Kubernetes ポートフォワードを介して行われます。

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

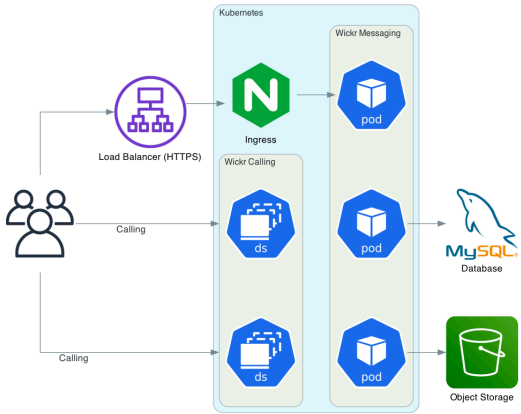
ライセンス要件

インストールには .yaml 形式のライセンスファイルが必要です。これは Wickr サポートから提供されます。

アーキテクチャ

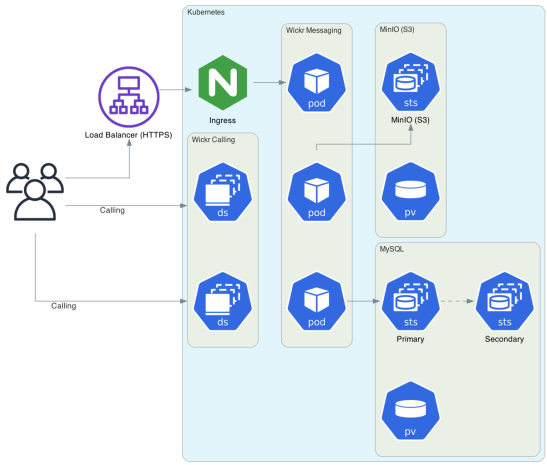
推奨される本番稼働用アーキテクチャ

次の図は、本番環境で推奨されている Wickr Enterprise と、Kubernetes クラスターの外部にある MySQL とオブジェクトストレージサービスの両方を示しています。



内部アーキテクチャまたはテストアーキテクチャ

次の図は、内部 MYSQL およびオブジェクトストレージサービスを利用した Wickr Enterprise の設定を示しています。特定のデプロイの特定のニーズを満たす場合がありますが、一般的な本番環境での使用はお勧めしません。



インストール

1. kubectl および kots CLI をインストールします。
2. Kubernetes クラスターに接続します。
3. Wickr サポートから Wickr Enterprise ライセンスファイルを取得します。
4. 次のコマンドを使用して Wickr Enterprise をインストールします。

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

> **Note**
>
> license.yaml は、提供されたライセンスファイルを表します。

初回インストール後、KOTS 管理コンソールはクラスターレベルの管理と設定オプションを提供します。

KOTS 管理コンソール

このインターフェースは Wickrエンタープライズのデプロイされたバージョンを管理するために使用されます。Wickr Enterprise のインストールステータスの確認、設定の変更、アップグレードの実行を行うことができます。KOTS 管理コンソールには Kubernetes ポート転送からのみアクセスできます。ポート転送は以下のコマンドで開くことができます。

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

イングレス設定

Ingress Controller

Wickr Enterprise は、次の 4 つのイングレスコントローラータイプをサポートしています。

- LoadBalancer (デフォルト)
  - ロードバランサーオブジェクトは、クラウドプロバイダーによって提供されることが多いにもかかわらず、完全にオンプレミスのインストールで明示的な設定が必要になる場合があります。
  - LoadBalancer サービスタイプで Ingress Controller (ingress-nginx) サービスをデプロイします。そのためには、Kubernetes クラスターが外部ロードバランサーをサポートするプラットフォームで実行されている必要があります。
- 既存の ALB
  - Ingress Controller を既存の ALB にアタッチします。
  - 既存の Application Load Balancer ターゲットグループ ARN を指定する必要があります。
- 既存の NLB
  - Ingress Controller を既存の NLB にアタッチします。
  - 既存の Network Load Balancer ターゲットグループ ARN を指定する必要があります。
- NodePort
  - Ingress Controller (ingress-nginx) は NodePort サービスタイプを使用するように設定されます。これにより、Kubernetes クラスター内のすべてのノードでポートが開き、トラフィックが Ingress に転送されます。クライアントトラフィックは、DNS または外部ロードバランサーを介してこれらのノードに送信できます。
  - ポート範囲は 1～65535 で選択できます。選択しない場合、30000-32767 のランダムポートが使用されます。
- Ingress

- 独自の Ingress Controller を使用します。この設定では、サービスが Ingress マニフェストで使用する Ingress クラス名を受け入れます。これは、イングレスコントローラーに他のロードバランシングメカニズムを介して設定済みの外部接続があることを意味します。
- 現在、ingress-nginx コントローラーのみがサポートされています。

ワイルドカードホスト名

デフォルトでは、イングレスルートはホスト値「*」で定義されます。Wickr Enterprise Server に定義されたホスト名を使用するには、この設定を無効にします。IP ベースのホスト名にはワイルドカードホスト名が必要です。

データベース設定

Wickr Enterprise には MySQL 8.0 データベースが必要です。MySQL 5.7 を使用している場合は、「」を参照してアップグレードMySQL 8.0 へのアップグレードしてください。Amazon RDS などの Kubernetes クラスターの外部にあるデータベースを使用することをお勧めしますが、インストールの一部として Kubernetes クラスター内に内部 MySQL データベースをデプロイすることもできます。

外部データベースの設定

- ホスト名: データベースサーバーのホスト名または IP アドレス。
- リーダーホスト名: データベースサーバーの読み取り専用エンドポイントのホスト名または IP アドレス (使用可能な場合) 。
- ポート: MySQL にアクセスするポート。
- データベース名: サーバーで作成されたデータベースの名前。
- ユーザー名: データベースへのアクセス許可を持つユーザー。
- パスワード: そのユーザーのパスワード。
- CA 証明書: TLS 経由でデータベースに接続するための PEM 証明書。

> **Note**
>
> MySQL のインストールで、デフォルトの latin1 文字セットと latin1_swedish_ci 照合が使用されていることを確認します。これは、MySQL サーバーが次のフラグで起動されたことを確認することで実現できます。

```
"--character-set-server latin1", "--collation-server
latin1_swedish_ci"
```

内部データベース設定

内部データベースタイプは、バイナリレプリケーションを使用する MySQL プライマリとセカンダリのクラスターに 2 つの StatefulSets をデプロイします。セカンダリはトラフィックを受信せず、ディザスタリカバリとバックアップにのみ使用できます。

ストレージサイズ: データベースポッドの永続ボリュームのサイズ (ギビバイト単位) 。

MySQL ストレージサイズの増加

> **Note**
>
> StorageClass のボリュームタイプは、ストレージサイズを増やすためにボリューム拡張をサポートしている必要があります。詳細については、「ボリューム拡張」を参照してください。

Wickr Enterprise で使用される MySQL サービスは、Kubernetes の StatefulSet リソースとしてデプロイされます。StatefulSets は、永続ボリュームクレームテンプレートなど、リソースの多くのプロパティをイミュータブルにします。StatefulSets のイミュータビリティの回避策として、MySQL で使用されるボリュームのサイズを増やすには、次のアクションを実行する必要があります。

1. `data-mysql-primary-0` および の永続ボリュームクレームを編集しますdata-mysql-secondary-0。
    1. `kubectl -n wickr edit pvc data-mysql-primary-0`. Set `spec.resources.requests.storage` 希望するストレージサイズにします。
    2. `kubectl -n wickr edit pvc data-mysql-secondary-0`. Set `spec.resources.requests.storage` 希望するストレージサイズにします。
2. 既存の StatefulSets を削除しますが、 `--cascade=orphan`フラグを渡してポッドを離れます。

    `kubectl -n wickr delete statefulset --cascade=orphan mysql-primary mysql-secondary`.
3. KOTS UI で、ステップ 1 で設定した値と一致するようにストレージサイズ設定を更新します。この設定を保存してデプロイします。

4. StatefulSets を再起動してボリュームを拡張し、MySQL サービスをオンラインに戻します。

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

MySQL 8.0 へのアップグレード

外部データベース (RDS)

Wickr バックエンドをオフラインにするには、次の手順を実行します。

1. Ingress の名前空間を検索する `kubectl get deployments --all-namespaces`

   以下の例では、名前空間は Wickr で、レプリカは 3 です。

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

2. Ingress をスケールダウンする `kubectl scale deployment/ingress-nginx-controller --replicas=0 -n wickr`
3. スナップショットを作成して DB をバックアップします。詳細については、「Amazon Relational Database Service ユーザーガイド」の「手動バックアップの管理」を参照してください。 Amazon Relational Database Service
4. エンジンバージョンを MySQL 8.0.x にアップグレードします (MySQL 8.4 はサポートされていません)。詳細については、Amazon Relational Database Service ユーザーガイド」の「DB インスタンスエンジンバージョンのアップグレード」を参照してください。

Wickr バックエンドをオンラインにするには、進入をスケールバックします `kubectl scale deployment/ingress-nginx-controller --replicas=3 -n wickr`

内部データベース

詳細については、MySQL のバックアップと復元」を参照してください。

S3 ファイルストレージ

Wickr Enterprise には S3 互換ストレージサービスが必要です。Amazon S3 などAmazon S3サービスを使用することをお勧めしますが、インストールの一部として Kubernetes クラスター内に内部 S3 サービスをデプロイすることもできます。

外部 S3 設定

- バケット名: ファイルのアップロードが保存される S3 バケットの名前。
- Region: S3 バケットの AWS リージョン。
- エンドポイント: Wickr が S3 API とやり取りするために使用するエンドポイントを設定します。デフォルトはリージョンの S3 サービスエンドポイントです。
- Fileproxy サービスアカウント名: Amazon S3 のみ。サービスアカウントの IAM ロールを使用した S3 への認証に使用する既存の Kubernetes サービスアカウントの名前。
- 外部 S3 アクセスキー: これは既存の S3 アクセスキーです。
- 外部 S3 シークレットキー: これは既存の S3 シークレットキーです。

内部 S3 設定

内部 S3 タイプは、それぞれ 4 つの永続ボリュームクレームを含む 4 つの MinIO サーバーポッドのデフォルトをデプロイします。デフォルト設定では、MinIO の消去コーディングを使用して耐障害性を高めます。

- 内部 S3 サーバー数: 作成する MinIO サーバーポッドの数。フォールトトレラントデプロイのデフォルトは 4 です。この値は、開発/テストデプロイでは 1 に設定できます。
- 内部 S3 ボリューム数: 各 MinIO サーバーポッドに作成する MinIO ボリュームの数。フォールトトレラントデプロイのデフォルトは 4 です。この値は、開発/テストデプロイでは 1 に設定できます。
- 内部 S3 ボリュームサイズ: MinIO サーバーポッドで作成された MinIO ボリュームのサイズ。デフォルトは 10GB です。
- デフォルトの内部 S3 デプロイでは、4 つの PVCs を持つ 4 つのサーバーが使用されます。各 PVC は 10 Gi で、160 Gi Raw ストレージと 120 Gi Erasure Coded ストレージをユーザーが利用できます。
- Minio Erasure Coding 計算ツールを使用できます。詳細については、「消去コード計算ツール」を参照してください。

永続ボリュームクレーム設定

Wickr Enterprise では、ステートフルデータを保存するために永続的なボリュームクレームが必要です。この設定では、使用するストレージクラスの名前を指定できます。空白のままにすると、Wickr はデフォルトのストレージクラスの使用を試みます。Wickr のデプロイ後のストレージクラスの変更はサポートされていません。

永続ボリュームクレームのデフォルトの StorageClass は、クラウドプロバイダーによって提供されることがよくありますが、完全にオンプレミスでのインストールでは、ロングホルンなどのサードパーティーサービスを使用した明示的な設定が必要になる場合があります。

TLS 証明書の設定

TLS を終了するための PEM 証明書とプライベートキーをアップロードします。証明書のサブジェクト代替名は、Wickr Enterprise デプロイの設定で設定されたホスト名と一致する必要があります。

証明書チェーンフィールドでは、アップロード前に中間証明書 (必要な場合) をルート CA 証明書と連結します。

Let's Encrypt

Let's Encrypt を使用して証明書を自動的に生成するには、このオプションを選択します。証明書は、cert-manager 演算子を介して HTTP-01 チャレンジを使用して発行されます。

HTTP-01 チャレンジでは、目的の DNS 名がクラスターのイングレスポイント (通常はLoad Balancer) に解決され、TCP ポート 80 へのトラフィックが一般公開されている必要があります。これらの証明書は有効期間が短く、定期的に更新されます。証明書を自動的に更新するには、ポート 80 を開いたままにする必要があります。

> **Note**
>
> このセクションでは、Wickr Enterprise アプリケーション自体で使用される証明書を明示的に参照します。

ピン留めされた証明書

Wickr Enterprise では、自己署名証明書またはクライアントデバイスで信頼されていない証明書を使用する場合、証明書のピン留めが必要です。Load Balancer によって提示された証明書が自己署名

であるか、Wickr Enterprise のインストールとは異なる CA によって署名されている場合は、ここで CA 証明書をアップロードして、代わりにクライアントにピン留めさせます。

ほとんどの場合、この設定は必要ありません。

証明書プロバイダー

Wickr Enterprise で使用する証明書を購入する予定の場合、証明書がデフォルトで正しく機能することがわかっているプロバイダーのリストについては、以下を参照してください。プロバイダーが以下にリストされている場合、証明書はソフトウェアで明示的に検証されています。

- Digicert
- RapidSSL

自己署名証明書の生成

Wickr Enterprise で使用する独自の自己署名証明書を作成する場合、以下のコマンド例には、生成に必要なすべてのフラグが含まれています。

```
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 ベースの自己署名証明書を作成する場合は、代わりに次のコマンドを使用します。IP ベースの証明書を使用するには、ワイルドカードホスト名フィールドが受信設定で有効になっていることを確認します。詳細については、「イングレス設定」を参照してください。

```
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**
>
> 例の $YOUR_DOMAIN を、使用するドメイン名または IP アドレスに置き換えます。

呼び出し設定

- ノードの呼び出しを要求する: この設定が有効になっている場合、Wickr の呼び出しサービスは というラベルの Kubernetes ノードにのみデプロイされますrole=calling。この設定を無効にして、呼び出しサービスとメッセージングサービスを同じノードにデプロイするか、単一ノードデプロイにデプロイします。

  また、TCP Proxy サービスはポート 443 で実行されるため、この設定が無効になっている場合は、通常、呼び出し元の TCP Proxy を無効にする必要があります。
- TCP プロキシを有効にする: この設定は、呼び出しで TCP フォールバックモードのサービスがデプロイされるかどうかを制御します。443/tcp で実行されている他のサービスがある場合、または呼び出しに TCP フォールバックモードを必要としない場合は、この設定を無効にします。これは、Wickr Open Access を使用する予定のデプロイで有効にする必要があります。
- サーバーのパブリック IP アドレスを自動的に検出する: この設定を有効にすると、呼び出し元のサービスは https://ipv4.icanhazip.com/と に HTTPS リクエストを送信してパブリック IP アドレスを検出しますhttps://ipv6.icanhazip.com/。無効になっている場合は、「トラフィックの呼び出しにホストプライマリ IP アドレスを使用する」または「ホスト名オーバーライド」設定を有効にする必要があります。有効にしないと、呼び出し元のサービスが開始されません。
- トラフィックの呼び出しにホストプライマリ IP アドレスを使用する: Kubernetes ノードのプライマリ IP アドレスを使用して サービスを呼び出します。これは、ダウンワード API status.hostIPから に示されているように、すべての Wickr クライアントがノードのプライマリ IP アドレスで Kubernetes ノードに接続できることを意味します。
- ホスト名の上書き: 呼び出しサービスの接続ポイントとして返すホスト名または IP アドレスを指定します。この設定は、サービスのすべてのレプリカに対して同じ値が返されるため、単一の呼び出しサーバーを実行する場合にのみ使用してください。ホスト名オーバーライドが設定され、「ホストプライマリ IP アドレスの使用」設定が有効になっている場合、ホストプライマリ IP アドレス設定が優先されます。
- ホストネットワークの呼び出しが有効: デフォルトでは、ポッドを呼び出すとノードのホストネットワークが接続に使用されます。これを無効にして、トラフィックを呼び出す NodePort サービスを公開します。イングレスの呼び出しが有効になっている場合は、イングレストラフィックを許可するように適切なサービスが設定されていることを確認してください。これは STIG コンプライアンスでは無効にする必要があります。

イングレス設定の呼び出し

Wickr は呼び出しイングレス設定をサポートしており、クライアントはクラスター内の呼び出しノードに接続し、正しい呼び出しサーバーへの呼び出しルートを持つことができます。Wickr は 4 つの呼び出しイングレスタイプをサポートしています。

- LoadBalancer (デフォルト)
  - LoadBalancer はクラウドプロバイダーによってプロビジョニングされます (完全にオンプレミスのインストールには追加の設定が必要です)。LoadBalancer がプロビジョニングされたら、KOTS 設定を再度更新して、ロードバランサーのホスト名または IP アドレスを指定する必要があります。
- NodePort
  - トラフィックを呼び出すエントリポイントとして機能する各呼び出しノードで NodePort サービスを公開します。1 つ以上のノードに解決されるホスト名、または 1 つ以上のノードの IP アドレスを指定する必要があります。UDP およびオプションで TCP トラフィックのポート範囲を 30000-32767 から選択できます。
- 既存の NLB
  - 呼び出し元の Ingress サービスを既存の NLB にアタッチします。UDP のターゲットグループ ARN と、オプションで TCP トラフィックを指定する必要があります。
- サービスなし
  - 進入トラフィックを許可するために追加の Kubernetes サービスが必要ない場合は、これを選択します。これは通常、ホストネットワーク設定で使用され、着信トラフィックを着信ノードに直接ルーティングします。

考慮事項

- Ingress を呼び出さずに古いクライアントやフェデレーティッドネットワークとの下位互換性を確保するために、Ingress を呼び出すことが有効になっている場合、レガシー呼び出しモードは引き続き使用できます (呼び出し元サーバーへの直接接続)。デフォルトポートを変更する場合は、呼び出し元のノードにポートの衝突がないことを確認してください。
- UDP トラフィックを処理するデュアルスタック NLBs には、IPv6 バックエンドターゲットが必要です。詳細については、「Network Load Balancer ターゲットグループ」を参照してください。
- STIG コンプライアンスが必要な場合は、 を呼び出すためのホストネットワークオプションを無効にする必要があります。ノードがデュアルスタックモードで設定されているが、クラスターがそうでない場合、IPv6 接続が失われる可能性があります (IPv4 クラスターを想定)。

- Ingress を呼び出すには、定義済みのホスト名または IP アドレスが必要です。ノードのスケーリングやカスタムルーティングの提供には、設定の変更が必要になる場合があります。
- デフォルトの呼び出しイングレスポートは、TCP の場合は 8443、UDP の場合は 16384 です。ファイアウォールとセキュリティグループがこれらのポートのトラフィックを許可し、デフォルトが上書きされている場合は代替ポートを許可していることを確認します。

リファレンスアーキテクチャ

ロードバランサーによるイングレス

このオプションは、単一のロードバランサーをすべての呼び出しトラフィックのエントリポイントとして公開します。

1. 着信タイプの呼び出しで、Load Balancerまたは既存の NLB を選択します。既存の NLB の詳細については、GitHub の Wickr Enterprise CDK サンプルで NLB スタックを参照してください。
2. 着信タイプに応じて、次のいずれかを実行します。

   - 既存の NLB の場合、UDP トラフィックと TCP トラフィックのターゲットグループ ARNs と NLB のホスト名を指定します。
   - Load Balancer の場合、Kubernetes によってプロビジョニングされた後にホスト名を指定します。

   または、着信タイプの呼び出しで、ロードバランサーの IP アドレスまたはロードバランサーを指すカスタムホスト名を指定することもできます。
3. (オプション) メッセージングと呼び出しトラフィックを単一の NLB で組み合わせるには、イングレスセクションで既存の NLB を選択し、HTTPS ターゲットグループを指定します。

NodePort を使用したイングレス

このオプションは、ホストネットワークが無効になっており、追加のロードバランサーを公開しない場合に便利です。

> **Note**
>
> ファイアウォールとセキュリティグループが NodePorts のトラフィックを許可していることを確認します。

1. 着信タイプの呼び出しで、NodePort を選択します。
2. 呼び出し元ノードのホスト名または IP アドレスを追加します。
3. 呼び出しホストネットワークを無効にします。

HostNetwork を使用した直接進入

このオプションは、追加の Kubernetes サービスを公開せず、イングレストラフィックを呼び出して、呼び出し元のノードのホストネットワーク経由で直接接続できるようにします。IPv6 接続が必要な場合は、このアプローチが推奨されます。

1. 着信タイプの呼び出しで、サービスなしを選択します。
2. 呼び出し元ノードのホスト名または IP アドレスを追加します。
3. ホストネットワークの呼び出しを有効にします。

Kubernetes クラスターオートスケーラー (オプション)

Kubernetes Cluster Autoscaler は、Wickr Enterprise インストールのオプション設定値です。これにより、トラフィックの増加やその他のリソース制限によりパフォーマンスが低下する可能性がある場合に、Kubernetes ノードグループのスケーリングに役立ちます。

Wickr Enterprise のインストールでは、、Google Cloud AWS、Azure の 3 つのクラウドプロバイダー統合がサポートされています。この統合には、クラウドプロバイダーごとに異なる要件があります。この機能を有効にするには、以下の特定のクラウドプロバイダーの指示に従ってください。

AWS

WickrEnterpriseCDK を使用して Wickr 環境をインストールしなかった場合は AWS、Cluster Autoscaler を有効にするための追加のステップを実行する必要があります。

1. ノードグループに次のタグを追加します。これにより、Cluster Autoscaler は適切なノードを自動検出できます。

    1. `k8s.io/cluster-autoscaler/`**`clusterName`** `= owned` ここで、clusterName は Kubernetes クラスターの名前です。
    2. `k8s.io/cluster-autoscaler-enabled = true`
2. Kubernetes サービスアカウントを kube-system 名前空間に追加し、自動スケーリングと ec2 アクションを許可する IAM ポリシーに関連付けます。詳細と詳細な手順については、「Amazon

EKS ユーザーガイド」の「IAM ロールを引き受けるように Kubernetes サービスアカウントを設定する」を参照してください。

1. サービスアカウントを設定するときは、「kube-system」名前空間を使用する必要があります。
2. サービスアカウントには、次のポリシーを使用できます。

   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"
        }
    ]
}
```

クラスターオートスケーラーを設定するときにレプリケートされた UI で、クラウドプロバイダーAWSとして を選択し、上記で作成したサービスアカウントの名前を指定して、そのサービスアカウントを使用するようにクラスターオートスケーラーに指示します。

Google クラウド

Autopilot クラスターと標準クラスターの両方で、GKE の組み込み Autoscaling 機能を使用することを強くお勧めします。ただし、この統合を続行する場合は、続行する前に次の要件を満たす必要があります。

要件:

1. マネージドインスタンスグループ (MIG) は、コンピューティングエンジンリソースに対する少なくとも「読み取り/書き込み」を含むセキュリティスコープで作成する必要があります。これは、後で MIG に追加することはできません。
2. クラスターでは、ワークロード ID フェデレーションが有効になっている必要があります。これを既存のクラスターで有効にするには、以下を実行します。 `gcloud container clusters update ${CLUSTER_NAME} --workload-pool=${PROJECT_ID}.svc.id.goog`
3. ロール「roles/compute.instanceAdmin.v1`. これは、以下の手順を使用して作成できます。

```
# 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) では、ほとんどのデプロイでクラスターの自動スケーリングが統合されているため、これらのメソッドをクラスターの自動スケーリングに使用することを強くお勧めします。ただし、これらのメソッドが機能しないような要件がある場合は、Azure Kubernetes Service 用の Kubernetes クラスターオートスケーラー統合を提供しています。この統合を利用するには、クラウドプロバイダーとして Azure を選択した後、次の情報を収集し、Cluster Autoscaler の下の KOTS 管理パネルの設定に配置する必要があります。

Azure 認証

サブスクリプション ID: サブスクリプション ID は、公式ドキュメントに従って Azure ポータルから取得できます。詳細については、Azure ポータルの「サブスクリプションとテナント IDs」を参照してください。

次のパラメータは、az コマンドラインユーティリティを使用して AD サービスプリンシパルを作成することで取得できます。

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

アプリ ID:

クライアントパスワード:

テナント ID:

Azure Cluster Autoscaler の設定

クラスターオートスケーラーを適切に機能させるには、認証要件に加えて、次のフィールドが必要です。この情報を取得するためのコマンドは便宜上提供されていますが、特定の AKS 設定によってはいくつかの変更が必要になる場合があります。

Azure マネージドノードリソースグループ: この値は、定義したリソースグループではなく、AKS クラスターを確立したときに Azure によって作成されたマネージドリソースグループです。この値を取得するには、クラスターの作成時に から CLUSTER_NAME と RESOURCE_GROUP が必要です。これらの値を取得したら、以下を実行してこれを取得できます。

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

アプリケーションノードプール VMSS 名: これは、Wickr アプリケーションの AKS ノードプールに関連付けられた仮想マシンスケーリングセット (VMSS) の名前です。これは、クラスターのニーズに基づいてスケールアップまたはスケールダウンするリソースです。この値を取得するには、次の az コマンドを実行します。

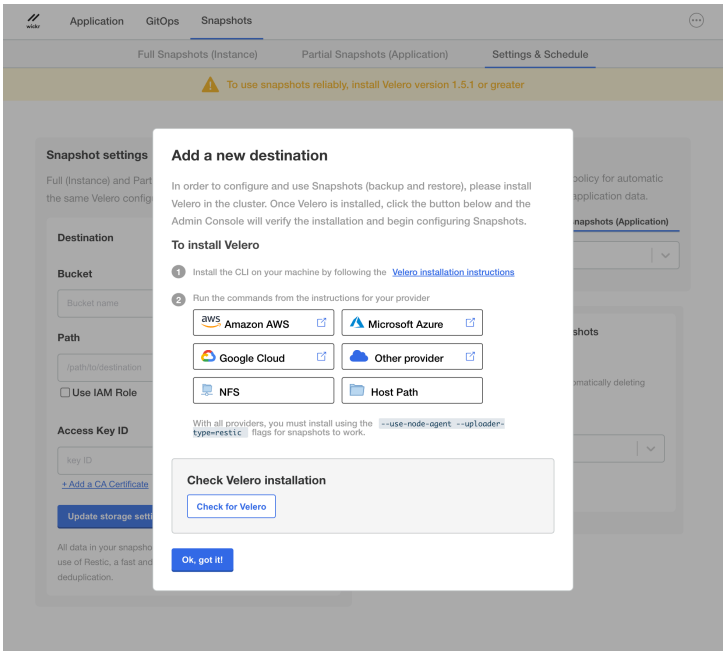
```
CLUSTER_NODEPOOL_NAME="(Your-NodePool-Name)"
CLUSTER_RESOURCE_GROUP="(Your-Managed-Node-Resource-Group-As-Defined-Above>)"
az vmss list -g ${CLUSTER_RESOURCE_GROUP} --query '[?tags."aks-managed-
poolName"==`'''${CLUSTER_NODEPOOL_NAME}'''`].{VMSS_name:name}' -o tsv
```

ACalling元ノードプール VMSS 名 (オプション): 呼び出し元ノードプールに関連付けられている VMSS がある場合は、その VMSS の名前です。この値を取得するには、呼び出し元のノードプールのノードプールの名前の CLUSTER_NODEPOOL_NAME 値を切り替えて、Application Node Pool VMSS Name の コマンドの修正バージョンを実行できます。

バックアップ

Wickr Enterprise はバックアップの目的で Velero を使用します。Velero は、クラウドプロバイダーでもオンプレミスでも、Kubernetes クラスターリソースと永続的ボリュームをバックアップおよび復元するために必要なツールを提供します。

Minio を使用した Velero バックアップ: 現在、Velero バックアップは Low Resource Mode の Minio でのみ有効になっています。



Velero ドキュメントを使用したインストール

- Velero CLI をインストールします。詳細については、「Velero CLI のインストール」を参照してください。
- クラスターに Velero をインストールし、プロバイダーに基づいてストレージを設定します。
  - AWS.
  - GCP。
  - Azure。
  - その他のプロバイダー。

制限

デフォルトでは、バックアップにボリュームは含まれません。バックアップする必要があるボリュームをポッドがマウントする場合は、バックアップに含める特定のボリュームを一覧表示する注釈を使用してバックアップを設定する必要があります。

バックアップが必要なボリュームごとに、backup.velero.io/backup-volumes 注釈を追加します。注釈名は backup.velero.io/backup-volumes で、値はバックアップに含めるボリュームのカンマ区切りリストです。詳細については、「スナップショットの設定」を参照してください。

Airgap のインストール

Wickr Enterprise と KOTS はどちらも、完全にエアギャップ化された Kubernetes クラスターへのデプロイをサポートしています。エアギャップ Kubernetes クラスターから到達可能なプライベート Docker イメージレジストリへのアクセスを提供する必要があります。KOTS に提供されるプライベート Docker イメージレジストリは、この目的のために正しく機能するためにユーザー名/パスワード認証で保護する必要があります。KOTS は Private Docker Image Registry を使用して、すべての Wickr Enterprise イメージをホストします。

- airgap が有効になっている Wickr Enterprise license.yaml (Wickr セールスまたはカスタマーサポートチームにお問い合わせください)
- Wickr Enterprise wickr.airgap アーカイブバンドル (Wickr セールスまたはカスタマーサポートチームにお問い合わせください)
- プライベート Docker イメージレジストリ へのアクセス。
- airgap 環境にデプロイされた Kubernetes クラスターへのアクセス。
- Kubectl がインストールされました。
- KOTS CLI がインストールされました。
- kotsadm.tar.gz をダウンロードしました。

次のコマンドを実行して、エアギャップ kubernetes クラスターに KOTS と Wickr Enterprise をデプロイします。これらのコマンドは、KOTS 管理イメージと Wickr Enterprise イメージを Private Docker Image Registry にアップロードします。コマンドが完了すると、上記のように KOTS 管理コンソールにアクセスして Wickr Enterprise のインストールを完了するように求められます。

```
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
```

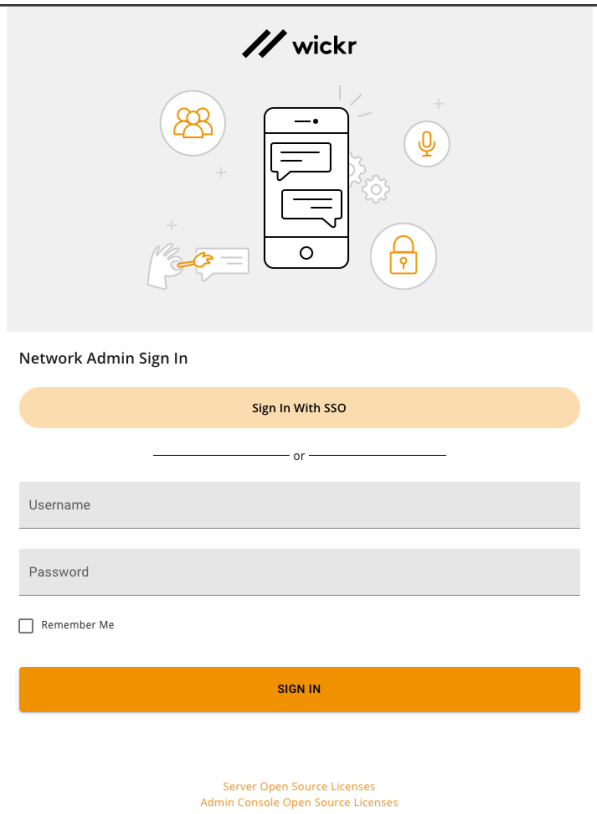
airgap インストールのモバイル通知

サーバーバックエンドからモバイルクライアントへのプッシュ通知には、追加のネットワーク許可リストが必要です。この要件は、Apple iOS と Google Android がオフラインデバイスとバックグラウンドデバイスにこの機能を実装する方法によるものです。これらのサービスのドキュメントを参照し、指定された IP アドレスとポートを許可リストします。

- iOS
- Android

Wickr 管理コンソール

Wickr 管理コンソールインターフェイスは、Wickr Enterprise アプリケーション自体を管理するために使用されます。ネットワーク、ユーザー、フェデレーションなどの設定に使用できます。ロードバランサーを指すように設定した DNS 名で HTTPS 経由でアクセスできます。デフォルトのユーザー名は admin で、パスワードは Password123 です。このパスワードは、初回ログイン時に変更する必要があります。



セキュリティ設定

AWS Wickr Enterprise には、デプロイのセキュリティコンテキストを強化するための設定が用意されています。この高いセキュリティ標準はポッドおよびコンテナレベルで適用され、セキュリティ技術実装ガイド (STIG) に準拠するために必要です。

次の設定パラメータを設定して、拡張セキュリティコンテキストを適用します。

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

> **Warning**
>
> Opensearch の場合、このセキュリティ設定により永続ストレージのアクセス許可を更新する `fsgroup-volume` initContainer が無効になり、アクセス許可に関連する互換性の問題が発生する可能性があります。

よくある質問

Q: helm stderr で次のエラーが発生してデプロイが失敗します。

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

A: これは、デバッグログ記録が有効になっている場合に発生する可能性があります。デバッグログ記録を無効にし、問題のあるジョブを削除して、もう一度試してください。

Wickr Enterprise 用の埋め込みクラスター

Wickr Enterprise の組み込みクラスターインストールオプションは、Wickr Enterprise 製品に小規模で効率的なインストールを提供します。レプリケートされた埋め込みクラスターを活用して、Wickr Enterprise をインストールできる k0 を使用した小規模な Kubernetes インストールを提供します。このインストール方法を使用すると、回復力と高可用性を犠牲にしてall-in-one」ソリューションを提供することで、Wickr Enterprise のインストールに関する技術的なスキル要件と全体的なハードウェア要件を最小限に抑えることができます。

トピック

Wickr Enterprise 埋め込みクラスターの開始方法

Wickr Enterprise 埋め込みクラスターオプションの使用を開始するには、 サポートに連絡してライセンスを受け取ってください。既存のライセンスがあり、このオプションを使用する場合は、既存のライセンスの更新と追加のインストール手順について サポートにお問い合わせください。

Wickr Enterprise 組み込みクラスターの要件

Wickr Enterprise 埋め込みクラスターのインストールを開始する前に、次の要件が満たされていることを確認してください。

ネットワーク要件

次のポートで Wickr サーバーへの進入を許可する必要があります。

- HTTPS の場合は 443/TCP

- TCP プロキシのみの呼び出し - KOTS での TCP 呼び出しトラフィック用に設定された TCP プロキシポート
- UDP 呼び出しトラフィックの 16384-19999/UDP
- LAN のみ - KOTS 管理コンソールにアクセスするための 30000/TCP

システム要件

インストールする前に、以下の最小リソースが利用可能な VM (仮想マシン) または Linux ベースのオペレーティングシステム (OS) を実行している物理マシンがあることを確認してください。

- 8 CPU コア
- 12 ギガバイト (GB) の RAM
- / (ルート) パーティション上の 100 ギガバイト (GB) のディスクストレージ

Wickr Enterprise 組み込みクラスターは、次の Linux OS システムでテストされていますが、他の Linux ベースの OS オプションも適している場合があります。

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

Wickr Enterprise 埋め込みクラスターのインストール (標準)

ダウンロード手順が完了したら、Wickr Enterprise バンドルを送信先マシンにダウンロードして解凍します。

```
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
```

これで、 `wickr-enterprise-ha`と の 2 つのファイルが作成されます`license.yaml`。`wickr-enterprise-ha` ファイルは、埋め込みクラスターのインストールに必要なすべての部分を含むバイナリファイルですが、 `license.yaml`はインストールの検証に使用される Wickr ライセンスです。

この段階で基本的なインストールを実行するには、 wickr-enterprise-ha ファイルを実行します。

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

インストールプロセスが開始されると、管理者コンソールのパスワードの入力を求められます。安全なパスワードを入力し、KOTS 管理コンソールにアクセスしてインストールの設定を続行するときに必要なとおりに保存してください。

インストールが完了すると、出力は次のようになります。

```
   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
```

標準インストール後、ウェブブラウザを使用して出力で提供される KOTS 管理コンソール URL に進みます。この例では、URL は ですhttp://192.168.1.100:30000。ただし、URL はネットワーク設定によって異なります。

マルチノードインストール

Wickr Enterprise Embedded Cluster マルチノードインストールは、埋め込みクラスターユーザーがの Wickr Calling ワークロードと Wickr Messaging ワークロードを異なる物理マシンに分離するオプションを提供します。これを行うために、Wickr Enterprise は レプリケートされた埋め込みクラスターマルチノードツールを活用します。

ポート要件

マルチノード機能を正しく動作させるには、クラスターのすべてのメンバーで次のポートを開く必要があります。これらはノード間で開くだけで、より広いインターネットには開かれません。

- 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

ライセンス要件

Wickr Embedded Cluster マルチノード設定オプションには、追加のライセンス権限が必要です。 サポート に連絡して、ライセンスがこの機能をサポートしていることを確認します。

初期設定時に追加のノードを作成する

Wickr Enterprise Embedded Cluster を最初に設定するときに、セットアッププロセス中に追加の呼び出しノードを作成できます。まず、「Wickr Enterprise 埋め込みクラスターのインストール (標準)」で説明されている手順に従います。KOTS 管理パネルに移動すると、追加のノードを作成するように求められます。

> **Note**
>
> 現在、埋め込みクラスターマルチノードは 1 つの呼び出しノードと 1 つのメッセージング/コントローラーノードのみをサポートしています。

開始するには、コントローラーロールオプションの選択を解除し、ロールの呼び出しオプションを選択します。これにより、新しいノードを設定するための追加の命令セットが入力されます。新しいノードでこれらの手順を実行して、クラスターを呼び出しノードとして結合するように設定します。

新しいノードで、次の例のような手順を実行します。

1. 新しいノードにバイナリをダウンロードします。

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

2. バイナリを抽出します。

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

3. ノードをクラスターに結合します。

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

join コマンドが正常に完了すると、呼び出しロールが割り当てられたクラスターの設定ページに新しいノードが表示されます。続行を選択して Wickr Enterprise 設定ページに進みます。KOTS 管理コンソール設定で説明されている埋め込みノード設定オプションの指示に従ってください。

既存の埋め込みクラスターインストールへのノードの追加

既存の Wickr Enterprise Embedded Cluster インストールに呼び出しノードを追加するには、KOTS 管理コンソールに移動します。これを行うには、ssh またはその他のメカニズムを使用してノードにログインし、インストールに使用されるwickr-enterprise-haバイナリを含むインストールディレクトリに移動します。`./wickr-enterprise-ha admin-console` を実行して KOTS 管理コンソールを起動します。このコマンドが出力を返さない場合、KOTS 管理コンソールはすでに実行されており、ウェブブラウザのノードの IP でポート 30000 に移動することでアクセスできます。例: `https://127.0.0.1:30000/`。

リクエストに応じて KOTS 管理者パスワードを入力し、次の手順を実行して追加のノードを作成します。

1. ログインしたら、KOTS 管理コンソールの左上にあるクラスター管理ページに移動します。
2. [Add node] (ノードの追加) を選択します。
3. でコントローラーの選択を解除しますRoles。

4. で呼び出しを選択する `Roles`
5. 表示される手順に従って、追加する新しいノードでコマンドを実行します。
6. 完了したら、閉じる を選択します。
7. 新しいノードが、呼び出しロールを持つノードリストに表示されます。
8. KOTS 管理コンソールの左上にあるアプリケーションページに移動します。
9. ページ上部のナビゲーションバーから Config を選択します。
10. 左側のナビゲーションパネルの呼び出しセクションに移動します。
11. 呼び出しノードの使用を許可するには、呼び出しノードを要求するを選択します。
12. ページの下部までスクロールし、設定の保存を選択します。
13. Config が更新されたことを示すポップアップが表示されます。Go to update version を選択します。
14. 更新されたバージョンページに、現在インストールされているバージョンが表示されます。新しい明細項目は、インストールされたバージョンの下に Config Change という名前で一覧表示されます。デプロイ を選択してこの新しいバージョンをデプロイし、新しい呼び出しノードを有効にします。

KOTS 管理コンソールの設定

KOTS 管理コンソールは、最初は自己署名証明書を使用します。これは、ブラウザで例外としてを許可する必要があります。この例外を受け入れると、KOTS 管理者コンソールの Configuration Wizard によって歓迎されます。このウィザードでは、必要に応じてカスタム証明書を追加するオプションなど、KOTS 管理コンソールの動作を設定するための追加の設定手順について説明します。

KOTS 管理者コンソールの初期設定が完了したら、インストールプロセス中に作成した管理者コンソールのパスワードを入力するように求められます。最初のログイン時に、クラスターを設定する必要があります。

続行 を選択して、Wickr の KOTS 管理コンソールに進みます。

単一ノードの埋め込みクラスターの場合は、続行を選択して Wickr の KOTS 管理コンソールに進みます。マルチノードのインストールについては、「マルチノードのインストール」を参照してください。

KOTS 管理コンソールで、必要に応じてインストールを設定します。組み込みクラスターサービスを利用する場合、Wickr Enterprise のインストールの適切な機能を確保するために設定する必要がある重要な構成設定がいくつかあります。

- ホスト名 - Wickr のインストールと通信するときに使用するホスト名です。Wickr Enterprise のインストールを指すには、このドメインに適切な DNS レコードを作成してください。
- 詳細オプションで、[ ] Ingress Controller の設定 オプションをチェックして、Kubernetes Ingress を設定するための設定ブロックを公開します。イングレス設定ブロック で、単一ノード埋め込みクラスターを選択し、ロードバランサー外部 IP (IPv4 のみ) というラベルのテキストボックスに Wickr サーバーに関連付けられた「パブリック」 IP を入力します。

  この IP が不明な場合は、Wickr サーバーのコマンドラインから次のコマンドを実行して、この値を決定できます。 `ip route get 1.1.1.1|awk '{print $7}'`
- 詳細オプションで、低リソースモードを有効にするオプションを確認します。
- 呼び出しで、単一ノードの埋め込みクラスターを使用している場合は、呼び出しノードの要求が選択されていないことを確認してください。それ以外の場合は、初期設定中に呼び出しノードを追加した場合は、呼び出しノードを要求するが選択されていることを確認してください。
- ファイル共有に外部データベースまたは S3 互換ストレージを使用しないオールインワンソリューションが必要な場合は、次の設定の内部オプションを選択します。
  - データベース
  - S3 ストレージの場所

    内部 S3 ストレージの場所には、ストレージ容量を設定するための追加オプションがあります。スケーリングはプロビジョニング後のオプションではないため、小規模から始めて必要に応じて拡張することをお勧めします。

必要な機能をすべて設定したら、設定ページの下部までスクロールし、設定の保存を選択します。これにより、いくつかのプリフライトホストチェックが開始されます。プリフライトチェックが完了したら、デプロイを選択して Wickr Enterprise Installation を開始します。

これで、Wickr Enterprise のインストールの設定を開始する準備が整いました。Wickr Enterprise の設定の詳細については、「Wickr Enterprise とは」を参照してください。

その他の一般的なインストール要件

IP ホスト名のインストール

インストールに IP ベースのホスト名が必要な場合は、追加の設定オプションがあります。これらの手順は IP ベースのホスト名に固有です。上記の基本的なセットアップについては、他の手順に従うことをお勧めします。

KOTS 管理パネルで、次の手順を実行します。

1. ホスト名を、使用する IP に設定します。
2. 証明書 で、証明書のアップロード を選択します。次に、IP ベースの証明書の手順に従って自己署名証明書を生成します。詳細については、「自己署名証明書の生成」を参照してください。
3. 証明書の .crt ファイルとプライベートキーの .key ファイルをアップロードする
4. Certificate Chain の場合は、.crtファイルを再度アップロードします。
5. 「ピン留めされた証明書を設定する」チェックボックスをオンにします。
6. ピン留めされた証明書.crtの をアップロードします。
7. 呼び出しで、「サーバーのパブリック IP アドレスを自動的に検出する」と「トラフィックの呼び出しにホストプライマリ IP アドレスを使用する」のチェックボックスをオフにします。
8. 呼び出しで、ホスト名の IP アドレスをホスト名オーバーライドテキストボックスに入力します。
9. 詳細オプションで、入力コントローラーの設定チェックボックスをオンにします。Ingress という新しい設定セクションが次に表示されます。
10. Ingress で、単一ノード埋め込みクラスターを選択します。
11. Ingress で、Wickr サーバーの「パブリック」インターフェイスの IP を入力します。これは、ホスト名として使用される IP とは異なる場合があります。この値の詳細については、基本的な設定手順を参照してください。
12. Ingress で、「ワイルドカードホスト名を使用する」を確認します。

SELinux 強制モード

強制モードで SELinux を使用する必要がある場合は、埋め込みクラスターのインストールに使用されるデフォルトのデータディレクトリを変更します。このユースケース/optでは、ほとんどの SELinux ポリシーで動作することがテストされているため、 を使用することをお勧めします。

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

レプリケートされた組み込みクラスターのデフォルトのインストール前チェックでは、SELinux が許容モードであることを検証しようとし、SELinux が強制モードの場合に失敗します。これをバイパス

するには、--ignore-host-preflightsコマンドライン引数を使用する必要があります。コマンドラインオプションを使用する場合、次のようなプロンプトが表示されます。プロンプトが表示されたら「はい」と入力します。

```
#  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 のインストール

Wickr Enterprise の組み込みクラスターインストールオプションは、エアギャップインストールをサポートしています。ライセンスには追加の設定と有効化が必要です。エアギャップ環境で Wickr Enterprise 埋め込みクラスターの使用に関心がある場合は、 サポートにお問い合わせください。

Airgap インストールを実行する場合、ダウンロード手順は標準のインストール方法とは異なります。次のようになります。

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

バンドルをインターネットアクセスが可能なマシンにダウンロードし、任意のデータ輸送方法を使用してエアギャップ環境に転送します。バンドルが転送されたら、標準のインストールバンドルと同じようにバンドルを抽出します。関連するすべての Wickr Enterprise アプリケーションサービスイメージwickr-enterprise-ha.airgapを含む 3 番目のファイル が含まれます。

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

インストール中は、抽出後に--airgap-bundleコマンドライン引数を設定する必要があります。設定されていない場合、プロセスは標準のインストール手順に従います。

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

airGapped 埋め込みクラスターの更新

AirGapped Embedded クラスターを更新するには、次の手順を実行します。

1. 新しい埋め込みクラスターパッケージを レプリケート からダウンロードし、エアギャップ環境の標準データ転送方法を使用してホストマシンに転送します。新しいバンドルがホストマシンに追加されたら、tarball を抽出します。

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

2. 新しいバイナリバンドルと airgap バンドルを使用して更新を実行します。

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

3. KOTS 管理コンソールを起動し、KOTS 管理コンソールにアクセスする標準的な方法を使用して、提供された URL にログインします。

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

4. KOTS 管理コンソールにログインしたら、バージョン の左にある最新の利用可能な更新を見つけ、バージョン履歴に移動ボタンを押します。
5. 利用可能な更新で、新しいバージョンのデプロイを選択します。画面を順を追って説明します。

   1. 任意の設定オプションを変更し、下にスクロールして次へを選択します。
   2. プリフライトチェックが失敗していないことを確認し、次へ: 確認してデプロイを選択します。
   3. [デプロイ] をクリックします。

Wickr Enterprise 埋め込みクラスターに関するその他の注意事項

- NAMESPACE: ほとんどの Wickr Enterprise のインストールとは異なり、組み込みクラスターのインストールでは、wickr ではなく kubernetes の kotsadm 名前空間に Wickr アセットがインストー

ルされます。kubectl、helm、またはその他のユーティリティ`-n wickr`に使用するスクリプトまたはコマンドを変更します`-n kotsadm`。

- Kubernetes クラスターの操作: ホストマシンから、 `./wickr-enterprise-ha`バイナリを使用して、 を実行して Kubernetes インストールとやり取りするように設定された適切な変数を持つシェルを作成します./wickr-enterprise-ha shell。これにより、シェルの PATH 内に kubectl ユーティリティが提供され、適切な kube 設定がローカルインストールに設定されます。

Wickr 埋め込みクラスターのインストールのトラブルシューティング

これらのトラブルシューティングステップのすべてのインスタンスは、Wickr Embedded Cluster のインストールを実行しているインスタンスへのシェルアクセスがあり、Kubernetes のインストールを直接操作できるように `./wickr-enterprise-ha shell` コマンドを実行していることを前提としています。

一般的な問題

クラスター管理画面から欠落しているノードボタンを追加する

Airgapped のインストール

エアギャップインストールを使用している場合は、この動作の修正について Wickr サポートにお問い合わせください。

標準インストール

ライセンスに埋め込みクラスターマルチノードの使用権限が含まれている場合は、ライセンス同期を実行して最新バージョンを取得します。この使用権限が不明な場合、または使用権限がない場合は、Wickr サポートにお問い合わせください。

ライセンス同期を実行するには、次の手順を実行します。

1. KOTS コントロールパネルに移動します。
2. ダッシュボードページで、ページの右上にあるライセンスセクションを見つけます。
3. このセクションでは、右上隅に Sync License ハイパーリンクが表示されます。ハイパーリンクを選択します。
4. ライセンスが同期されると、UI 更新と数秒前の最終同期が表示されます。

5. KOTS ダッシュボードページのバージョンセクションから再デプロイを選択します。
6. 再デプロイが完了したら、クラスター管理に戻り、ノードを追加できます。

アップグレードの問題

クラスターのアップグレード時にアップグレードがスタックする

アップグレードがクラスターのアップグレードで停止した場合、一部のポッドが適切に終了されていない可能性があります。インスタンスにログオンし、 `./wickr-enteprise-ha shell` コマンドを使用して kubernetes のインストールを管理するシェル環境に入ります。

1. まだ実行中のポッドを識別します。

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

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

   > **Note**
   >
   > 実行中のポッドの 1 つが `embedded-cluster-upgrade-XXXXXXXXXXXXXX-xxxxxkotsadm-xxxxxxx`または である場合は、アップグレードの実行にこれらのポッドが必要なため、削除しないでください。

3. 実行中のポッドが残っていないことを確認します。

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

この手順では、クラスターのアップグレードを Wickr アップグレードで続行できます。

クラスターのアップグレード中にアプリケーションが更新されず、新しいバージョンをデプロイできない

アップグレード後もアプリケーションが古いバージョンのままである場合、新しいバージョンは一貫性のない状態になる可能性があります。

Kubernetes のインストールレコードを確認します。

1. インストーラから Kubernetes シェルを開きます。

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

2. 次の kubectl コマンドを実行します。

```
kubectl get installations
```

3. 出力は次のようになります。

```
      [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. 失敗したインストールを削除します。

```
kubectl delete installation 20251113180133
```

5. KOTS 管理パネルを使用してアップグレードを再度実行しようとします。

RabbitMQ Pod が ログ行で失敗する **Error while waiting for Mnesia tables: {timeout_waiting_for_tables}**

RabbitMQ シークレットとストレージが同期されていません。これは通常、複数の RabbitMQ インスタンスが実行され、リーダーの選択またはクォーラムエラーが発生した場合に発生します。これを修正するには、RabbitMQ サービスとそのストレージボリュームを削除してから再デプロイします。

失敗した RabbitMQ を削除するには、次の手順を実行します。

1. RabbitMQ ステートフルセットを削除します。

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

2. 残りの RabbitMQ ポッドを削除します。複数の RabbitMQ-X ポッドが実行されている場合は、このコマンドを複数回発行して、追加のポッド名に対応するように RabbitMQ-X 値を更新します。

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

3. 対応する PVCs を削除します。実行中のポッドが複数ある場合は、適切なポッドに対応するように data-RabbitMQ-X を複数回更新して、このコマンドを発行します。

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

4. ポッドが残っているかどうかを確認します。成功すると何も出力されません。

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

5. 残りの PVCs があるかどうかを確認します。成功すると何も出力されません。

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

6. KOTS 管理パネルを使用して再デプロイします。

トラブルシューティングの詳細については、「トラブルシューティング」を参照してください。

ドキュメント履歴

次の表に、Wickr Enterprise 自動インストールガイドのドキュメントリリースを示します。

変更	説明	日付
セキュリティ設定	セキュリティ設定が追加されました。詳細については、「セキュリティ設定」を参照してください。	2025 年 8 月 26 日
マルチノードインストール	マルチノードインストールが追加されました。詳細については、「マルチノードのインストール」を参照してください。	2025 年 8 月 26 日
イングレス設定の呼び出し	イングレス設定の呼び出しが追加されました。詳細については、「イングレス設定の呼び出し」を参照してください。	2025 年 8 月 26 日
自動デプロイオプション	自動デプロイオプションが追加されました。詳細については、「Wickr Enterprise のインストール」を参照してください。	2024 年 2 月 23 日
許可リストへのポート	ポート TCP/8443 が許可リストに追加されました。詳細については、「Requirements」を参照してください。	2024 年 2 月 12 日
許可リストへのリソースとポートの破棄	リソースを破棄する手順が追加されました。詳細については、「リソースの破棄」を参照してください。さらに、許	2023 年 8 月 17 日

	可リストへのポートが追加されました。詳細については、「Requirements」を参照してください。	
初回リリース	Wickr Enterprise 自動インストールガイドの初回リリース	2023 年 8 月 4 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。