自分のコンピュータを設定する方法
このラボでは、Chrome ブラウザを使用します。

このラボの作業を行うには、事前設定済みの dCloud ラボを使用するか、自分のコンピュータを設定する必要があります。詳細については、「Pre-Event Preparation（イベント前の準備）」および「Lab Setup（ラボの設定）」の各モジュールを確認してください。

NeXt UI ツールキットの概要

このラーニング ラボでは、NeXt UI ツールキットを使用したヒューマン インタラクションの基本について学びます。dCloud のラボ環境で、ブラウザに NeXt UI を導入し、その機能について確認していきます。NeXt のフレームワークを使用して、カスタマイズした独自のトポロジを構築する方法についても確認します。このラボを終えることで、NeXt UI ツールキットの利点に加え、同ツールキットを利用して独自のアプリケーションを構築する上での基本についても把握することができます。

目的

所要時間：30 分

- NeXt UI による可視化がもたらす利点について理解する
- NeXt UI ツールキット使用の基本を理解する
- NeXt を使用するさまざまなプロジェクトについて理解する
- NeXt を使用してネットワーク トポロジを可視化する

前提条件

- このラボでは、ラボの設定を最小限にするために、dCloud のラボ環境を使用します。ラボ環境で自分のローカル ラップトップまたは Ubuntu ホストを使用して演習を行う方法については、「Lab Setup（ラボ設定）」および「Pre-Event Preparation（イベント前の準備）」の各モジュールを確認してください。
- 自身のコンピュータ上で作業をする場合は、ギット リポジトリを複製してください。詳細については、「**How to Setup Your Own Computer（自分のコンピュータを設定する方法）**」を参照してください。

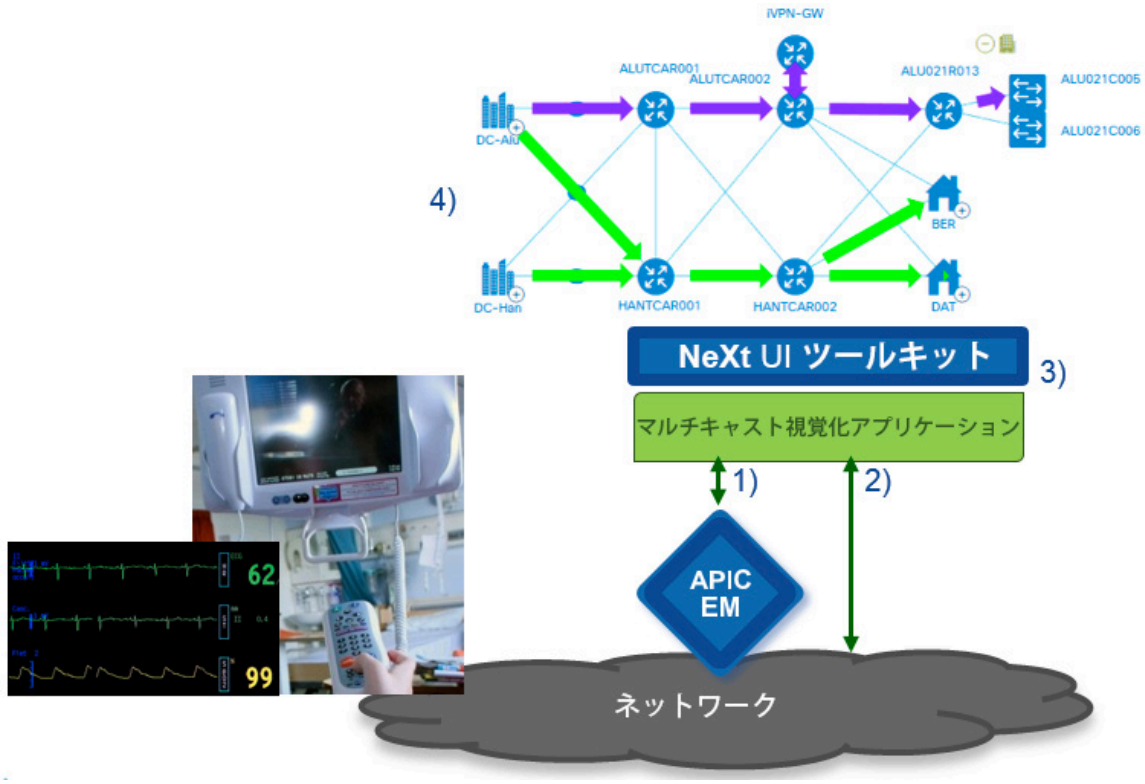
ステップ 1：NeXt UI ツールキットの利点について理解する

NeXt UI ツールキット（Network Embedded Experience）の目的は、ネットワークトポロジを可視化し、ネットワーク エクスペリエンス全体を向上させるための柔軟なソフトウェア フレームワークを提供することです。ネットワークのエンジニアリングと運用は、高度な視覚化が必要なプロセスです。ネットワークは、複数のプロトコル レイヤから構成されていることが多く、コマンド ライン インターフェイス（CLI）の出力を読み解いて理解するよりも、図によって把握する方が簡単です。トポロジ図がない状態で、ネットワークのトラフィックの流れを詳細に理解できるか想像してみてください。エンジニアやオペレータは、特定の設定や実装の意図を理解する上で、トポロジ図や設計ドキュメントを必要とします。このような背景を考慮すると、ネットワーク オペレータやエンジニアには日常的に対処せざるを得ない以下のような課題があることがわかります。NeXt UI ツールキットはそれらの課題を解決できます。

1. 日々更新されるトポロジの視覚化
    - エンジニアは大抵の場合、Visio や OmniGraffle のダイアグラムを通してネットワークトポロジを理解しています
    - これらのダイアグラムは、通常、手動で更新されており、情報が古くなってしまうことがあります
    - もし、ダイアグラムが最新のネットワーク実装状況を反映していなければ、運用上の問題となるでしょう
2. ナレッジの共有
    - 通常のトラブルシューティングや監査において、ネットワークの微妙な変化やプロトコル レイヤの状況は、複数のチームメンバーによって検討される必要があります
    - ダイアグラムは、時に複雑なネットワークの詳細を共有するための優れた手段です

たとえば、現場における実際の課題の例として、次のようなシナリオについて考えてみます。病院で、さまざまなメディア共有機器や重要な医療機器に対してマルチキャスト フォワーディングを行っていると仮定します。ここで、ネットワークでのマルチキャスト フォワーディング パスの詳細について、他のエンジニアと検討する場合を想像してみてください。上位レベルでのマルチキャストは単純ですが、実際にマルチキャストを詳細にトラブルシューティングしようとすると、非常に複雑になるはずです。マルチキャスト フォワーディングのトポロジを把握するには、IGMP や PIM の知識に加え、多くのマルチキャストの概念についても理解する必要があります。さらに、必要な情報を抽出するには、デバイスの CLI の特徴についても把握しなければなりません。

NeXt UI ツールキットを使うことで、こうした課題に対処することができます。下の図で示すように、NeXt は APIC-EM などのプラットフォームや個々のネットワーク デバイスからの情報を使用し、物理ネットワーク ダイアグラム上の特定のグループに関する既存のマルチキャスト転送トポロジを表示します。このような視覚化された情報が、それを必要とする運用担当者に共有され、トラブルシューティングやナレッジの共有、場合によってはコンプライアンスにも使用されます。



上記の例は、NeXt UI ツールキットのソリューションの全体像について示したものですが、ソリューションの各詳細については、次のような内容となります。

1. ネットワーク トポロジは、APIC-EM REST API を使用したカスタム アプリケーションから取得されます
2. マルチキャスト フォワーディング トポロジは、各ネットワーク デバイスから取得されます
3. プロトコルの状態とトポロジの情報に基づいて、NeXt UI ツールキットが、シンプルなダイアグラムを生成します。ダイアグラムでは、パスを視覚化するために、特定のグループに関するマルチキャスト転送パスが物理的なトポロジの上に重ねて表示されます
4. 運用担当者は、NeXt UI をシンプルな Web サービスとして共有することができます

これは、NeXt UI を使用したネットワーク情報の可視化の一例です。ラーニング ラボの次のステップでは、NeXt UI ツールキットに関する詳細について説明します。

ステップ 2：NeXt UI ツールキットのコンポーネントについて理解する

ここまでで、NeXt UI ツールキットを利用することで解決できる問題について学びました。次に、NeXt の仕様を確認し、独自のアプリケーションに統合する方法についてみていきましょう。NeXt の目標は、ネットワークを中心としたカスタム アプリケーションに対して、柔軟なグラフィカル UI を提供することです。シスコの APIC-EM など、さまざまなプラットフォームで、インベントリや物理ネットワーク トポロジなどのネットワークの詳細を取得する API が提供されています。NeXt を使用することで、先の例で示したようなマルチキャストの複雑な状況を、シンプルなトポロジのビューで表示できます。NeXt では、Visio や PPT で手作業でダイアグラムを作る必要がなく、更新や管理も簡単になります。

NeXt の概要

NeXt は、シスコ社内から始まったオープン ソース プロジェクトです。ソース コードは、ギットハブのリポジトリで閲覧できます。また、DevNet の Next UI ポータルからダウンロードすることもできます。

NeXt の中核となっているのは、HTML5、CSS、JavaScript のフレームワークです。これらのフレームワークにより、NeXt は AngularJS などの他の Web フレームワークと統合できるようになり、クライアント アプリケーションがブラウザを使用して NeXt のトポロジを共有できるようになります。

NeXt UI ツールキットは多くの場面で採用されるようになってきており、APIC-EM や OpenDayLight などの他の複数のプラットフォームやプロジェクトで、トポロジの可視化に使用されています。

NeXt UI によるトポロジの可視化の基本

NeXt を使用した基本的なトポロジ作成においては、次の 4 つの大まかな手順があります。

1. 最初に、NeXt UI のパッケージやライブラリをインストールする必要があります。この作業には、`bower` や `npm` などのパッケージ マネージャを使用できます。また、Angular.js などの Web フレームワークをインストールすることによってクライアント アプリケーションを利用することもできます。

2. 2 番目に、必要な NeXt のコードがインストールされた状態で、NeXt の共通トポロジ モデル（CTM）を使ってトポロジを作成する必要があります。共通トポロジ モデルは、トポロジの構造を記述するもので、NeXt UI ツールキットが、JavaScript を使用してトポロジを表示させる際に必要になります。JavaScript に関する知識があれば役に立ちますが、動作させる上で必須なものではありません。たとえば、「Coding 102 lab Step 6（コーディング 102：ラボの設定）」では、JavaScript でのコーディングをせずに、NeXt UI と APIC-EM を統合させる事例について紹介しています。
3. 3 番目に、NeXt UI のインスタンスを生成するためのアプリケーション ロジックを構築する必要があります。具体的には、NeXt UI で使用されるアイコンであるとか、アプリケーション ウィンドウのサイズなどの細部を構成する JavaScript を作成する必要があります。
4. 最後に、2 番目、3 番目の手順で作成した JavaScript のコードを呼び出すための基本的な HTML を用意する必要があります。そうすることで、アプリケーションがブラウザで閲覧可能な状態となります。

ラーニング ラボの次のステップでは、これまでのステップについてさらに詳しい内容を説明し、NeXt UI ツールキットを使ってカスタム アプリケーションを構成する方法について理解を深めてゆきます。

ステップ 3：NeXt の共通トポロジ モデルの基本

ここまでで、NeXt アプリケーションを開発する際のワークフローの概要について把握しました。次に、NeXt の基本的なトポロジを作成します。次の 2 つのラーニング ラボのステップでは、トポロジ モデルを作成してブラウザで表示させることを目標とします。

NeXt の共通トポロジ モデル

前のスライドでは、NeXt が共通トポロジ モデル（CTM）と JavaScript を使用することについてふれました。共通トポロジ モデルのファイルをアプリケーション用に編集することで、独自のカスタム トポロジを表示させることができます。

共通トポロジ モデルの基本を理解するために、まず `devnet-express-code-samples` リポジトリのあるディレクトリに移動します。同ディレクトリは「**ラボの設定**」モジュールで指示されたように、dCloud ラボ内もしくは自身のコンピュータに複製されています。dCloud ラボ環境の場合、コードのサンプルは Ubuntu サーバのホーム ディレクトリにあります。

このディレクトリには、3 つのファイルがあるはずです。

```
$ cd mycode/devnet-express-code-samples/module08/03-environment-03-mission/08-human-interaction-02-overview-of-next
.
.
.
$ ls
Data.js          Shell.js         my-topo-01.html
.
.
.
```

各ファイルは、次に示すようにそれぞれ異なる役割を持っています。

- `Data.js` は、前のステップで説明した共通トポロジ モデルです。以降のラーニング ラボでは、基本的なトポロジ ダイアグラムを作成するためにトポロジ モデルを編集する方法を説明します。

- `Shell.js` は、前のステップで説明したアプリケーション ロジックにあたります。このコードは本ラーニング ラボ用に作成されたもので、更新する必要はありません。
- `my-topo-01.html` は、前のステップで説明した JavaScript を呼び出すための HTML コードです。このコードも、本ラーニング ラボ用に作成されたもので、更新する必要はありません。

では、ラーニング ラボを開始する前に、共通トポロジ モデルについて 1 行ずつみていきましょう。以下は、`Data.js` ファイルの内容となります。

```
var topologyData = {
    nodes: [
        {"id": 0, "x": 410, "y": 100, "name": "router 1"},
        {"id": 1, "x": 660, "y": 100, "name": "router 2"}
    ],
    links: [
        {"source": 0, "target": 1}
    ]
};
```

このコードの動作について確認していきます。

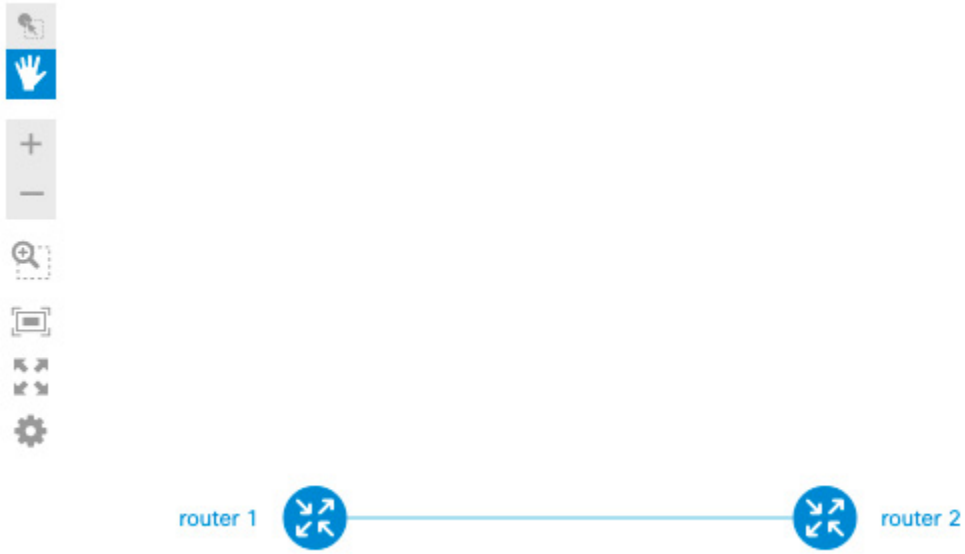
- まず、`topologyData` と呼ばれる JavaScript のオブジェクトを作成します。このオブジェクト名は、トポロジの定義の際に、NeXt UI ツールキットで使用されます。
- 次に、トポロジで 2 つのノードを定義します
    - 1 番目のノードには、`router 1` という名前と ID「`0`」が割り当てられます。さらに、NeXt UI トポロジ上での `x`、`y` 座標も指定されます。
    - 2 番目のノードには、`router 2` という名前と ID「`1`」が割り当てられます。NeXt UI トポロジ上での `x`、`y` 座標も指定されます。
- そして、トポロジのこれらのノードを接続するリンクを定義します
    - 今回は、ノードが 2 つしかないため、それらを接続するリンクが 1 つだけ必要になります。
    - リンクのソースはノード `0` で、ターゲットがノード `1` になります。

通常、`toplogyData` オブジェクトには、3 つのプロパティが設定されます。

- `nodes`:ノードはトポロジ ダイアグラム内のオブジェクトになります。`nodes` プロパティは、個別のオブジェクトの配列によって定義されます。たとえば、`nodes` には、ルータ、スイッチ、キャンパス ロケーションなどが含まれます。
- `links`:リンクはノードを接続します。`links` プロパティは、複数の個別のオブジェクトによって定義されます。
- `nodeSets`:ノードセットにより、個々のノードを 1 つのコンテナに格納できます。`nodeSets` がどのように機能するかは、別途、説明します。

共通トポロジ モデルの各プロパティに設定されている属性値を更新することで、フレームワーク内での NeXt のトポロジ表示を変更できます。

では、共通トポロジ モデルについて確認したところで、トポロジがどのように表示されるかみてみましょう。今回の例では、作業を容易にするために、NeXt アプリケーションを定義する JavaScript と、その JavaScript を呼び出す HTML についてはすでに作成済みです。Chrome ブラウザのウィンドウから、`my-topo-01.html` という名前の HTML ファイルを開きます。NeXt UI ツールキットにより、次のトポロジが表示されます。



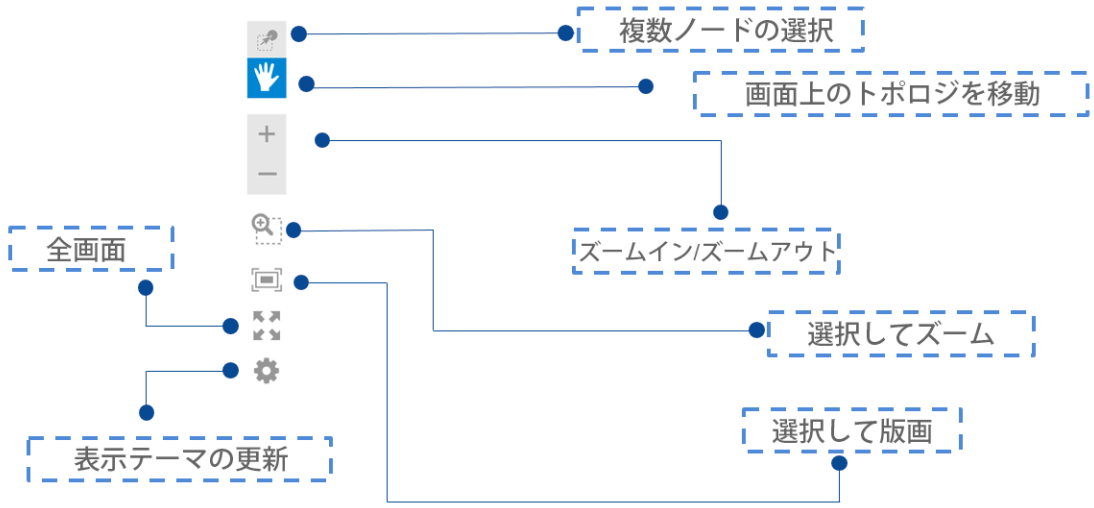
ここまでで、NeXt は、複数のネットワークトポロジを表示することができる汎用的なフレームワークだということを説明しました。ラーニング ラボの次のステップに進み、NeXt UI ツールキットのその他の機能について確認してください。

ステップ 4：NeXt UI のナビゲーション

ここまでで、NeXt の基本的なトポロジのモデリング方法について確認しました。このラーニング ラボのステップでは、NeXt UI ツールキットの他の機能についてみていきます。具体的には、UI のナビゲーションに関する内容と、共通トポロジ モデルの更新方法について説明します。

NeXt UI のナビゲーション

NeXt UI には、便利な機能がいくつか組み込まれています。Chrome ブラウザでトポロジを立ち上げた後、以下に紹介する機能を確認できます。



また、UI ウィンドウの中で、NeXt のトポロジ アイコンをドラッグ アンド ドロップして、移動させ見やすい表示へ修正することもできます。

NeXt のノード セットを使用した共通トポロジ モデルの更新

前のスライドで、ノードセットを使えば、各ノードをまとめることができると説明しました。この機能は、トポロジ ダイアグラムを階層化しようとする場合に役立ちます。たとえば、キャンパス サイトが 2 つあり、それぞれに複数のネットワーク デバイスがあるとします。NeXt UI では、それぞれのサイトを、各ノードを格納したノードセットとして表示させることができます。

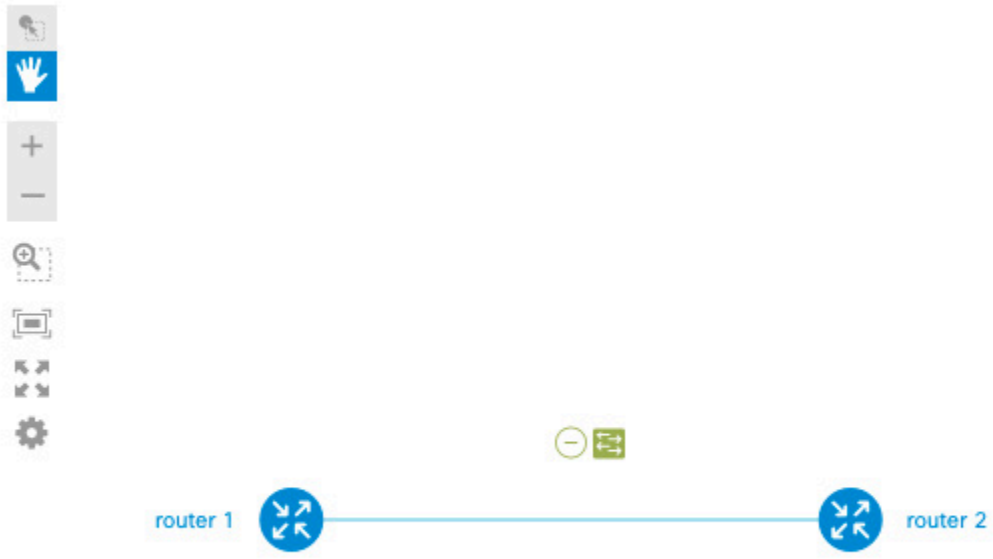
こうした概念について説明するために、共通トポロジ モデルの更新方法についてみていきましょう。`Data.js` ファイルの内容を、ノードセットの定義を含んだ下のデータに置き換えます。必要な内容は次のとおりです。

```
var topologyData = {
    nodes: [
        {"id": 0, "x": 410, "y": 100, "name": "router 1"},
        {"id": 1, "x": 660, "y": 100, "name": "router 2"}
    ],
    links: [
        {"source": 0, "target": 1}
    ],
    nodeSet: [
        {"id": 0, "nodes": [0, 1]}
    ]
};
```

上記のコードでは、新しく `nodeSet` を追加して、その固有の `ID` として、「`0`」を割り当てています。そして、新しく作成した `nodeSet` にノード `0` と `1` をマッピングしています。

更新された NeXt トポロジを表示させるには、ブラウザを更新してください。下の図にように表示されるはずです。

このように、NeXt UI によりノードセットが作成され、個々のノードを格納するスイッチ アイコンが表示されました。ではここで、スイッチ アイコンの横にある「+」アイコンをクリックしてみてください。ノードセットが展開されます。



ここまでで、JavaScript を使用した共通トポロジ モデルで、トポロジのモデリングができることについて説明しました。共通トポロジ モデルにより、ノード、リンク、ノードセットの各プロパティを利用して、ダイアグラムが構成できるようになっています。

次のラーニング ラボではこのモジュールに関するミッションを開始し、Spark、Tropo、NeXt を使用して複雑なヒューマン インタラクションをさらに確認していきます。

完了