自分のコンピュータを設定する方法
このラボでは導入部分を扱います。

環境のセットアップ
このモジュールの目標は、*DevNet Express for DNA* ラーニング トラック用にラップトップを準備することです。

目標

完了時間：15 ～ 60 分（設定するシナリオにより異なります）

- 準備するシナリオの決定
- Python がインストールされていることの確認
- ネットワーク接続の確認
- アカウントへの登録（Cisco Spark など）
- 必要なツールが利用可能であることの確認

前提条件

なし

ステップ 1：設定するシナリオの決定

次の 5 つのシナリオが考えられます。準備するシナリオは、ラボの環境や使用する開発環境、またハンズオン イベントへの参加の有無によって異なります。

状況に合わせて、最適なシナリオを選択してください。

シナリオ	設定内容	最低限準備すべき内容
#1	ハンズオン イベントに参加し、dCloud 内の開発ツールを使用する	• 次のいずれかのインストール • Chrome や Firefox などの HTML5 をサポートするブラウザ、または • AnyConnect および RDP クライアント • CCO アカウントの取得
#2	ハンズオン イベントに参加し、ローカル マシンで開発ツールを使用する	• AnyConnect のインストール • CCO アカウントの取得 • 接続テストの実行 • ローカル マシンに必要なツールのインストール
#3	自身の環境から参加し、dCloud 内の開発ツールを使用する	• Chrome や Firefox などの HTML5 をサポートするブラウザのインストール、または • AnyConnect および RDP クライアントのインストール • CCO アカウントの取得 • 接続テストの実行

シナリオ	設定内容	最低限準備すべき内容
#4	自身の環境から参加し、ローカル マシンの開発ツールを使用する	• AnyConnect のインストール • 接続テストの実行 • ローカル マシンに必要なツールのインストール
#5	自身の環境から参加し、ローカル マシンと自身のラボ環境を使用する	• 必要なツールのインストール • 所有しているルータ、コントローラ、ネットワークインフラの準備

次のページでは、それぞれのシナリオの準備に必要な手順について詳しく説明します。

開発ツールは、ローカル マシン、またはリモートの dCloud ワークステーションのどちらで使用するべきでしょうか。

ローカル マシン

ローカル マシンを使用する理由は何でしょうか。ラップトップなどのローカル マシンで開発ツール（Python、Postman など）を実行することもできます。受講後にすべての開発ツールを使用できるようにしたい場合は、ローカル マシンを使用すると便利です。マシンで使用中のツール（Python、Postman など）の数に応じて、設定には 1 時間程度かかります。過去にこれらのツールをインストールしたことがない場合は、少し難しいと感じることもあります。

ローカル マシンを使用する場合、コードの構築および実行は、マシン上で行われます。ローカル コードが dCloud のリモート インフラストラクチャ（APIC-EM やルータなど）と「通信」する際は、VPN 接続が必要となります。

リモート dCloud ワークステーション

dCloud ポッドでホストされるラボでも、ツールを使用できます。この設定はよりシンプルで、所要時間も少なくて済みます。dCloud ポッドにはすでにラボ環境が設定されており、Windows ワークステーションや Linux ワークステーションが用意されています。これらのワークステーションにはラボの際に必要なツール（Python、Postman など）が備わっています。dCloud ポッドへのアクセスには、次の 2 つの方法があります。

- AnyConnect（VPN）を使用した dCloud ポッドへのアクセス、リモート デスクトップか VNC クライアントを経由したワークステーションへのアクセス、または
- HTML5 をサポートするブラウザ（Firefox または Chrome）を使用した、dCloud ポッドのワークステーションへのアクセス

この場合、コードはリモート ワークステーション（Windows または Linux）で構築されます。コードはリモートで実行され、同一リモート ネットワーク内で実行されることから、インフラストラクチャと直接「通信」できます。VPN は必要ありません。

手順については、dCloud ポッドへのアクセス方法（IP アドレス、ユーザ名/パスワードなど）のトラックで説明します。このトラックの開始には、これらの情報は必要ありません。

ラボ環境について

どのラボ環境を使用すべきでしょうか。

通常は、DevNet がホストするラボ環境を使用します。ラボ環境は、シスコの dCloud ポッドでホストされます。このトラック用の dCloud ポッドには、各 VM（APIC-EM コントローラ、VIRL ネットワーク シミュレーション、IOS-XE ルータ、ワークステーションなど）と、このラボで学習するサーバ サイドのツールがまとめられています。

また、自身のラボ環境を使用して、このトラックのラボ実習を行うこともできます。すでにネットワーク デバイスと APIC-EM コントローラを含むラボ環境がある場合は、自身のラボを使用することができます。その場合は、自身で設定したコントローラとルータを準備する必要があります。

この学習トラックの内容にこれまで触れたことがない場合は、DevNet がホストする dCloud のラボを使用することを推奨します。必要なものがすべて含まれており、設定も少なくて済みます。

次の手順

設定の詳細については、次のページで説明します。次のページでは、自身のシナリオに適した手順を実行します。

自分のコンピュータを設定する方法
このラボでは導入部分を扱います。

ステップ 2：追加ツールのインストール

このページに記載されているツールは、ラボを完了させる上で有用なものです。設定によっては、すでにツールの一部を利用している可能性があります。

シナリオ 1 を設定している場合（リモート デスクトップ経由で dCloud ポッドの開発ツールを使用する場合）は、このページに記載のツールは必要ありません。このページを省略して、ステップ 5 へ進んでください。

Cisco Spark

（必須）ご利用のプラットフォーム用に、Cisco Spark クライアントをダウンロードおよびインストールしますネイティブ クライアントは、Windows および Mac、または Android および iOS で使用可能です。Linux の場合、またはネイティブ クライアントをインストールしない場合は、Web クライアントを使用することができます。

Chrome

（オプション）Chrome は HTML5 対応ブラウザです。ローカル マシンのデスクトップから dCloud ポッドへのアクセスに使用する場合はインストールします。

Postman

（必須）Postman は REST API のテストや確認を行うための**ツール**です。スタンドアロン バージョンもしくは Chrome の拡張機能としてインストールします。

RDP クライアント

（オプション）リモート デスクトップ クライアントをインストールします。すべてのプラットフォーム向けにさまざまな選択肢が利用可能です。dCloud ポッドを使用する場合や、ブラウザ ベースの RDP セッションを使用しない、またはできない場合に必要となります。

テキスト エディタ

（必須）ソース コード ファイルの編集の際に、テキスト エディタが必要となります。プラットフォームや個人の環境設定に応じて、大きく異なるため、推奨要件としてご検討ください。使用するテキスト エディタが、テキスト ファイルをバイナリ形式ではなくテキスト形式で保存できることを確認してください。

- **Windows**：Atom、Sublime Text、Notepad++ を選択できます。特に希望がなければ、動作が軽く無料のオプションがある NotePad++ がよいでしょう。
- **Mac OS X**：この場合も、Atom、Sublime Text が推奨されます。他のオプションとしては、TextWrangler が挙げられます。
- **Linux**：この場合も Atom、Sublime Text が利用でき、また gedit や Kate などのさまざまなオプションも選択できます。

Git クライアント

(**必須**)Git ソース コードの管理クライアントも、サンプル コードや他のソフトウェアを複製する際に必要になります。グラフィカル フロント エンドにはさまざまなオプションがあります。また、CLI のみを使用することもできます。主要なプラットフォーム用の無料クライアントが用意されている git-scm.com のダウンロード サイトや、あらゆる種類のプラットフォームに対応する代替/サードパーティ GUI クライアントの一覧も利用できます。

また、https://desktop.github.com から GUI バージョンをインストールすることもできます。

インストールの際には、Git ツールにパスが通っていることを確認してください。また、パス/環境への変更はすぐには反映されないことにも注意してください。新しくシェルを開くか、コンピュータを再起動する必要があります。

```
C:\Users\userid>echo %PATH%
C:\Python35\;C:\Python35\Scripts\;C:\Python27\Scripts;C:\Windows\system32;C:\WinndowsPowerShell\v1.0\;C:\Program Files (x86)\PuTTY\;C:\Program Files (x86)\WindoFiles\Git\cmd


C:\Users\userid>
```

上記の出力は、Git にパスが通っていることを示します(最後のエントリ)。

SSH/SCP

(**必須**)最初から SSH が含まれていない場合(Linux や Mac OS X ユーザの大半は該当しないはずです)は、SSH スイートのインストールを推奨します。Windows では Putty がよく使用されています。

もう 1 つの強力な Windows ツールとして、WinSCP があります。Windows との親和性が高く、簡単で安全なファイル転送が実現します。

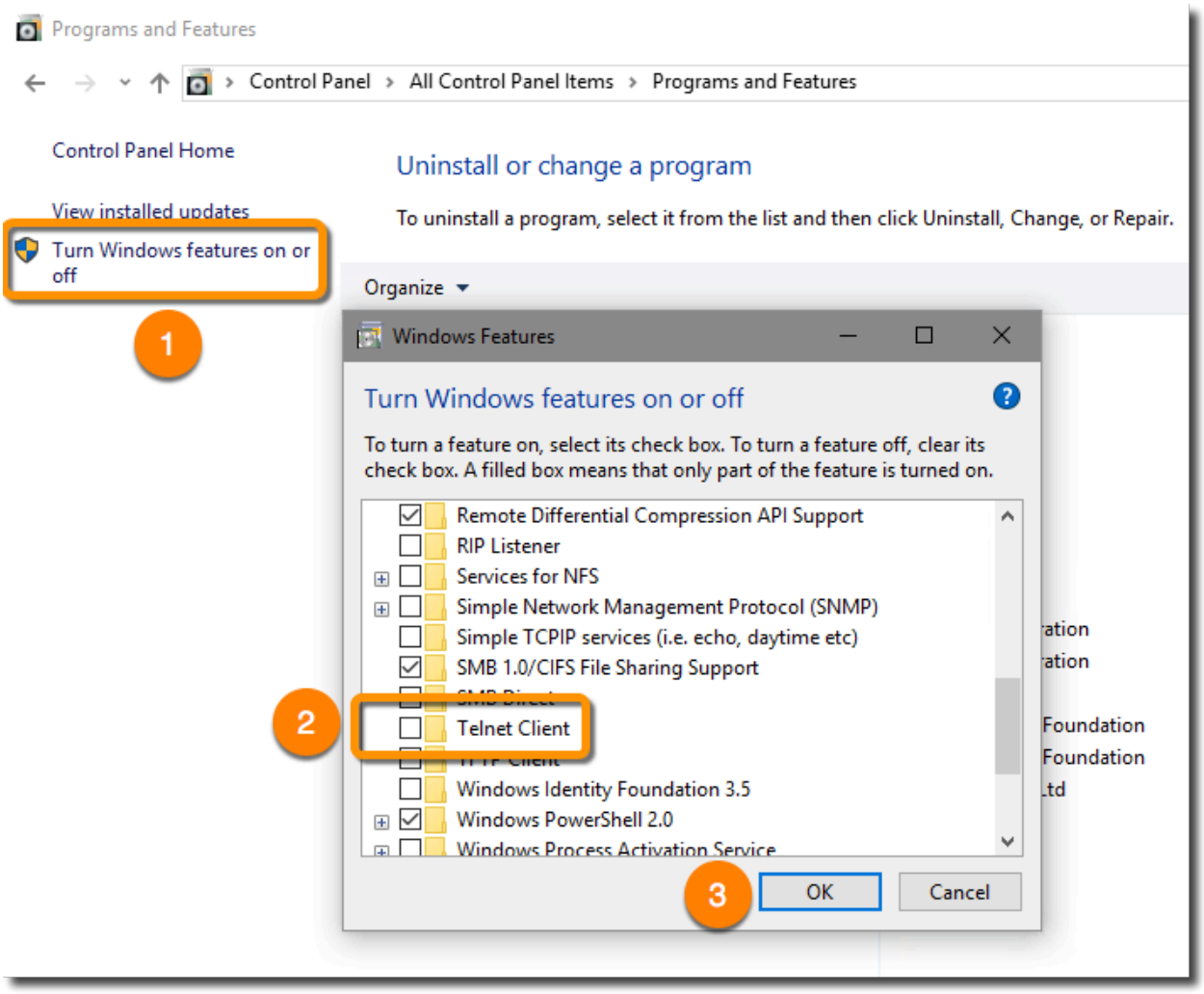
ZIP ツール

(**推奨**)任意の ZIP ツールをインストールします。7-Zip や Winzip が、Windows ではよく使用されています。Mac ユーザであれば、デフォルトでインストールされた `zip` や `unzip` を使用できます。Linux ユーザであれば、これらのツールを手動でインストールする必要があることもあります。

```
$ apt-cache search zip | grep -e '^zip -' -e '^unzip -'
unzip - De-archiver for .zip files
zip - Archiver for .zip files
$
```

Telnet

(オプション)聞き間違いではありません。リモート コンピュータのサービスへの接続をテストする際に、Telnet はいまだに有用なツールです。Telnet では必要最小限のブラウザと接続のテスト機能を利用できます。Mac OS X にはデフォルトでインストールされています。Windows では、Windows のインストール メディアからインストールする必要があります。



また、Linux では、ほとんどの場合でデフォルトのインストールには含まれず、手動でインストールする必要があります。

```
$ apt-cache search telnet | grep '^telnet -'
telnet - The telnet client
$ sudo apt-get install -y telnet
```

Cygwin

(オプション) Cygwin *は、Windows 上で Linux ディストリビューションと同等の機能を提供する GNU とオープン ソース ツールの広範囲なコレクションです。*完全にオプションですが、Cygwin ではたくさんの強力な CLI ツールが提供されており、プログラマーとして *nix 系のノウハウを利用したり学習したりする際に、その作業を容易にしてくれます。

Wireshark

(推奨) ラボの中には、パケット キャプチャ ツールを必要とするものがあります。tcpdump（Mac および Linux）、もしくは本格的な GUI ツールである Wireshark を使用できます。Wireshark はすべてのプラットフォームで使用でき、https://www.wireshark.org/ からダウンロードできます。

次のステップ

シナリオ 2、4、5 向けに、ローカル マシンに開発ツールをインストールする場合は、次のページに進んで Python 環境のインストールに関する情報を取得してください。それ以外の場合は、ステップ 4 に進みます。

自分のコンピュータを設定する方法
このラボでは導入部分を扱います。

ステップ 3：ローカル マシンへの開発ツール（Python、PIP など）のインストール

シナリオ 2、4、5 向けに、ローカル マシンに開発ツールをインストールする際は、このページの手順を完了する必要があります。

ローカル マシンに開発ツールをインストールしない場合は、このページを省略して次のページのステップ 4 に移動してください。

ご利用の OS によっては、すでに Python 環境がインストールされている場合もあります。Mac OS X の場合は、Python がインストールされています。また、ほとんどの Linux ディストリビューションには Python がインストールされています。ラボの例では、Python のバージョン 2.7 および 3.x に互換性があります。

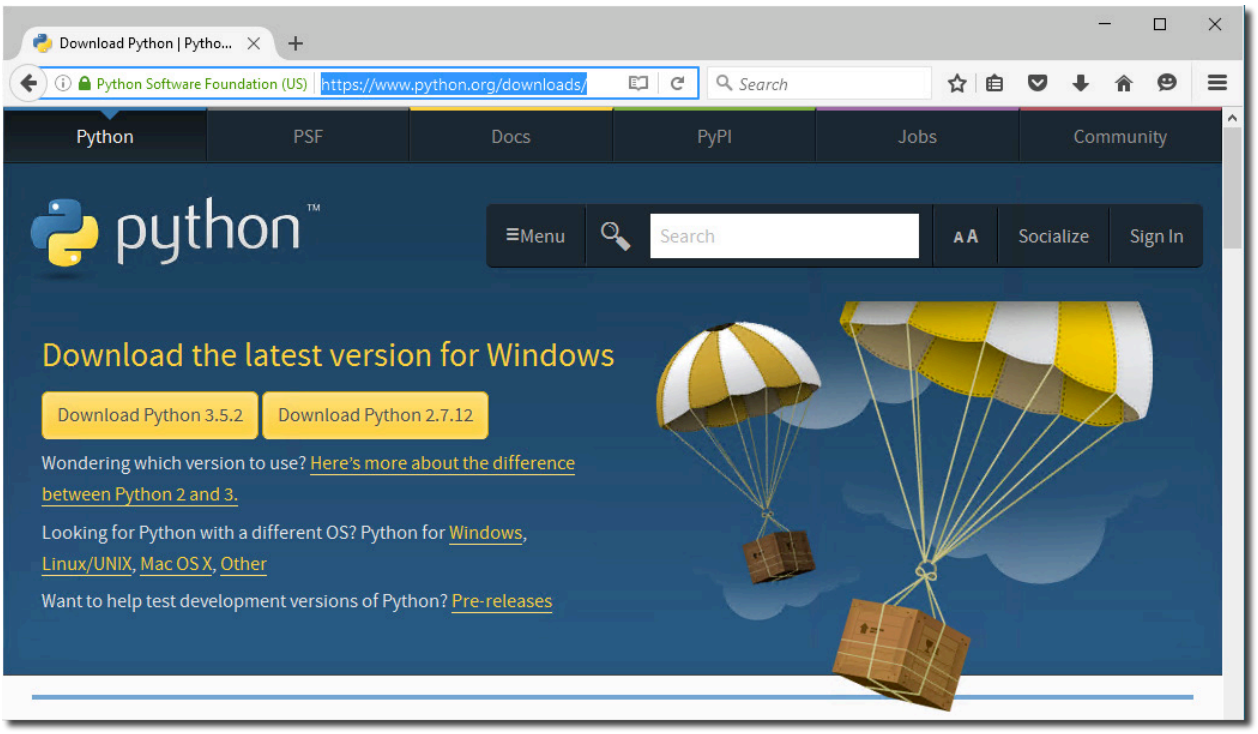
Python がインストールされていない場合は、次の手順に従ってインストールします。

Python がマシンにあることを確認した後、さらに下の **PIP** および**仮想環境**のセクションを参照します。これらのセクションでは、必要な Python パッケージのインストール方法、および Python の仮想環境の構築方法について説明します。Python の仮想環境を用意することは、ベスト プラクティスの 1 つとして考えられます。それによりグローバルな Python 環境への*影響を防ぐ*ことができるためです。

Python インストーラの入手

Python のインストーラをダウンロードします。

1. https://python.org に移動します
2. ダウンロード ページに移動します。



設置場所

いくつかのオプションがあります。

- コンピュータへのネイティブ インストール
- コンテナ内へのインストール（たとえば、Docker 環境を持っている場合）
- VM 内へのインストール（迅速な立ち上げと稼動という点では、Vagrant が適したツールです）
- 手動でインストール、維持する VM（VirtualBox や VMware Workstation/Fusion など）内にインストール

これらのアプローチには長所と短所があります。疑問に思う点がある場合は、マシンに最適なアプローチについて同僚や講師に確認してください。

Windows OS でのインストール

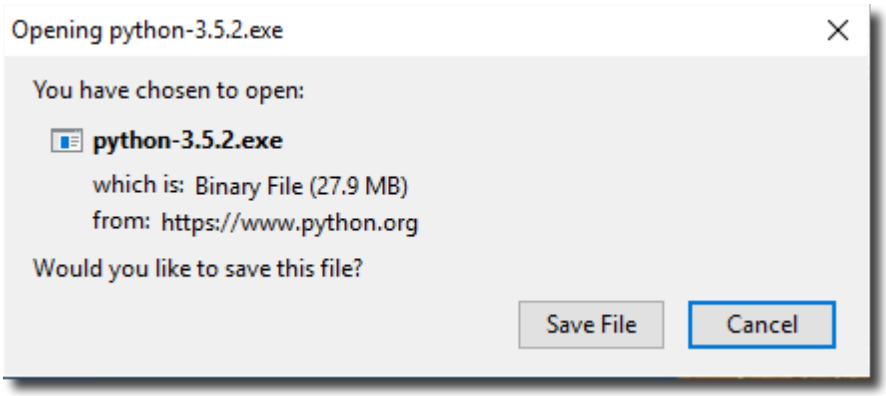
オペレーティング システムのバージョンに一致するバージョンをインストールします。たとえば、64 ビット バージョンの Windows を実行している場合は、Python の 64 ビット バージョンをインストールします。

複数の Python 環境を実行する場合（たとえば、バージョン 2.7 と 3.5 を並行して実行する場合）は、コンピュータのハード ドライブのルートにインストールすることで、パスを容易に適用できます。たとえば、「`C:\Python27`」や「`C:\Python35`」などのディレクトリにインストールします。

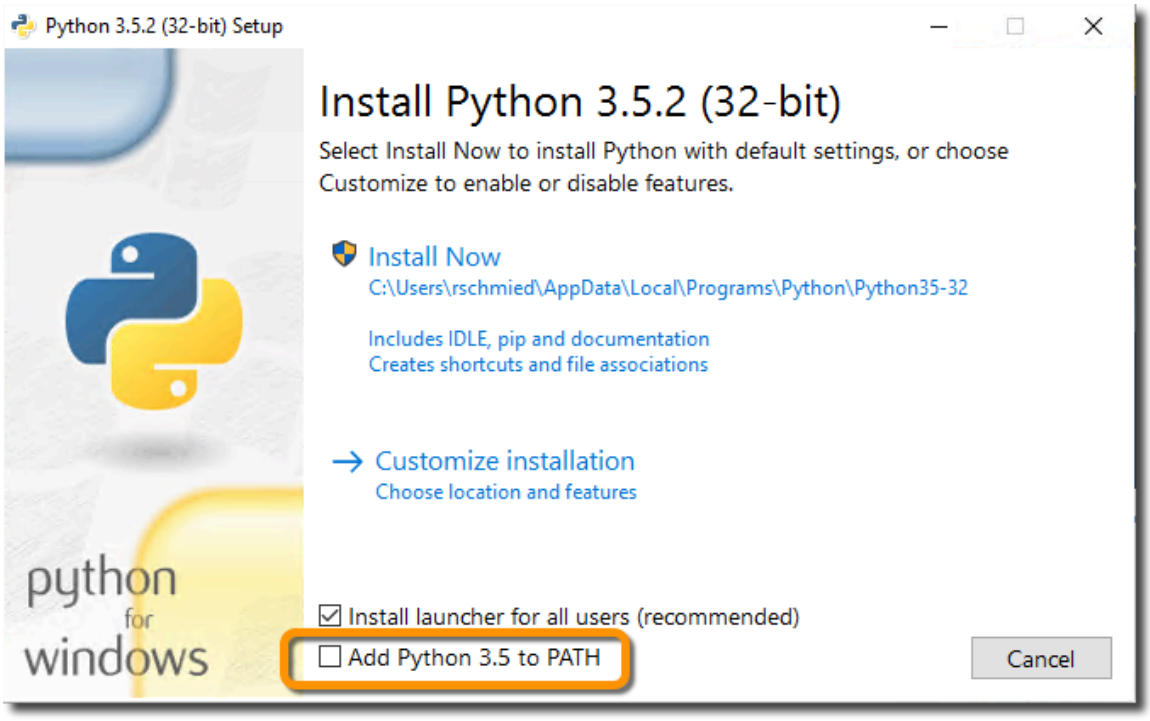
インストーラで表示される最初のダイアログで、[カスタマイズ インストール（場所と機能を選択）（Customize installation（Choose location and features））] を選ぶことで、インストール時のディレクトリ名を変更できます。

[インストール（Install）] をクリックする前に、カスタマイズ シーケンスにある [Python を環境に追加（Add Python to the environment）] チェックボックスをオンにします。

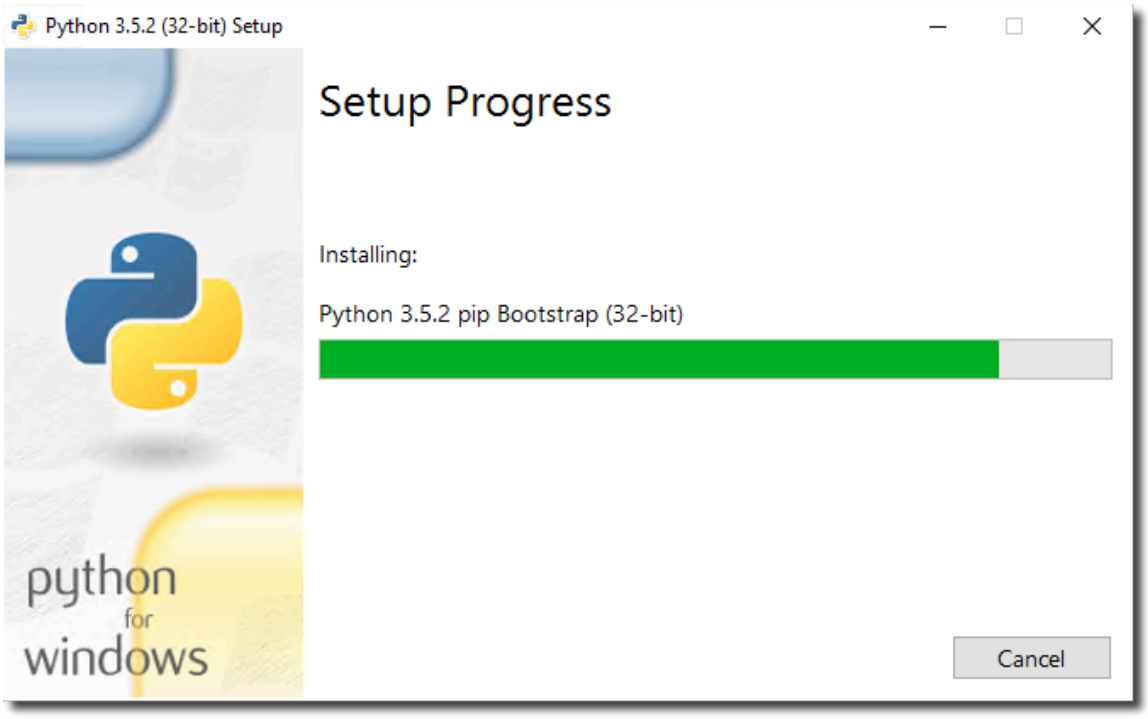
1. インストーラをダウンロードして実行します



2. [パスに Python 3.5 を追加（Add Python 3.5 to PATH）] ボックスをオンにします。



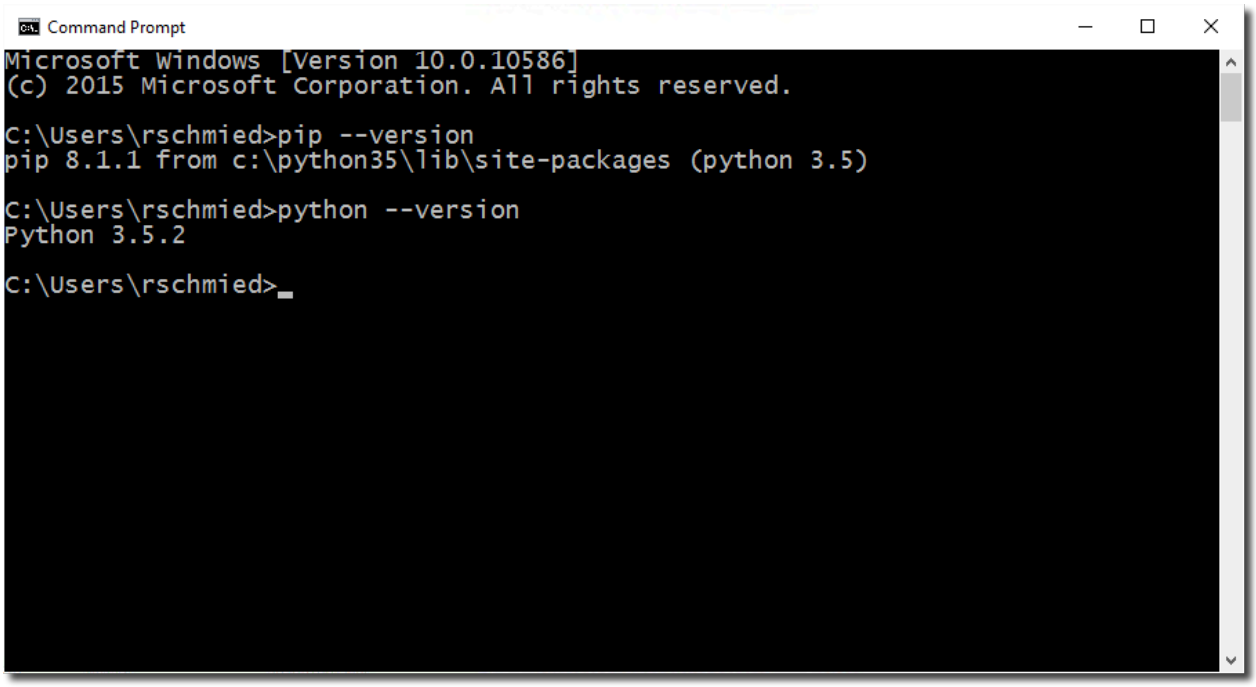
3. [今すぐインストール（Install Now）] をクリックしてインストールを完了します。



4. インストーラを閉じます。

5. コマンド ウィンドウを開きます（`cmd.exe`）
6. コマンドラインで、「`pip --version`」と入力して、PIP のインストールを確認します。

7. コマンドラインで、「`python --version`」と入力して、Python のインストールを確認します。



Mac および Linux でのインストール

通常、Mac OS X や Linux では、Python 環境がインストール済みとなっています。どちらのプラットフォームでも、Python バージョン 2.7 がデフォルトです。Mac OS X を利用中で、Python 3.5 にアップグレードする場合は、https://python.org に移動します。インストール パッケージをダウンロードします。

Linux を使用している場合は、パッケージ マネージャを使用して、「python3」パッケージを見つけます。通常は、プラットフォームのパッケージ マネージャに含まれているため、https://python.org のサイトに *nix 系プラットフォームのバイナリ リリースはありません（プラットフォームのパッケージ マネージャとしては、Ubuntu の `apt` や RPM ベース システムの `yum`/`dnf` があります）。

Python が正常にインストールされていることを確認します。

1. コマンドライン ウィンドウを開きます（ターミナル）。
2. 「`python --version`」と入力します。

次の例は、Mac OS X にインストールされた Python 2.7 の一般的な出力を示します。

```
$ python --version
Python 2.7.10
$
```

正確なバージョンは、インストールのソースに応じて異なります。

- OS X 付属の Python
- Homebrew または MacPorts、または、
- https://python.org からのバイナリ パッケージ

Python 3.5 をインストールした場合は、次のような出力になります。

```
$ python --version
Python 3.5.2
$
```

複数の Python バージョンを同時にインストールした場合は、希望する Python のバージョンを確実に使用できるよう、パスを適宜変更するようにしてください。また、このドキュメントも参照してください。

すべてのプラットフォーム

ローカル マシンに Python をインストールした後、次の説明に沿って、Python 環境に追加のパッケージをインストールします。

Uniq パッケージ

Uniq は、シスコの Application Policy Infrastructure Controller Enterprise Module（APIC-EM）のノースバウンド API 用の Python API クライアント ライブラリです。2016 年 7 月まで、APIC-EM が Uniq パッケージと連携するためには、Python のバージョン 2.7.x が必要でした。パッケージは現在、Python のバージョン 3.5.x でも使用できます。

PIP のインストール

`pip` ツールは、Python パッケージ インデックス PyPI のインターフェイスとなる Python パッケージ マネージャです。Python パッケージのインストールを管理する際は、この pip を使用することを推奨します。Python パッケージは Python のモジュールのコレクションです。モジュールをインポートすることで、Python 環境に有用な機能を追加することができます。

Windows OS での PIP

Windows に Python をインストールした場合、`pip` はインストールの一部として導入されています（インストールの際に無効にしていない場合）。コマンド ウィンドウで次を入力することで、インストールされているかテストができます。

```
pip -version
```

Mac OS X での PIP

Mac OS X では、`pip` がインストールされていない可能性があります。PIP がインストールされているか確認します。

1. コマンド ライン（ターミナル ウィンドウ）で、次を入力します：`pip --version`

```
$ python --version
Python 2.7.10
$ pip --version
-bash: pip: command not found
$
```

7. コマンドがみつからない場合、PIP をインストールする必要があります。

```
$ sudo easy_install pip
Password:
Searching for pip
Reading https://pypi.python.org/simple/pip/
Best match: pip 8.1.2
Downloading https://pypi.python.org/packages/e7/a8/7556133689add8d1a54c0b14aeff0acb03c64707ce100ecd53934da1aa13/pip-8.1.2.tar.gz#md5=87083c0b9867963b29f7aba3613e8f4a
Processing pip-8.1.2.tar.gz
Writing /tmp/easy_install-mEZelf/pip-8.1.2/setup.cfg
Running pip-8.1.2/setup.py -q bdist_egg --dist-dir /tmp/easy_install-mEZelf/pip-8.1.2/egg-dist-tmp-VZMCVd
warning: no previously-included files found matching '.coveragerc'
warning: no previously-included files found matching '.mailmap'
warning: no previously-included files found matching '.travis.yml'
warning: no previously-included files found matching '.landscape.yml'
warning: no previously-included files found matching 'pip/_vendor/Makefile'
warning: no previously-included files found matching 'tox.ini'
warning: no previously-included files found matching 'dev-requirements.txt'
warning: no previously-included files found matching 'appveyor.yml'
no previously-included directories found matching '.github'
```

```
no previously-included directories found matching '.travis'
no previously-included directories found matching 'docs/_build'
no previously-included directories found matching 'contrib'
no previously-included directories found matching 'tasks'
no previously-included directories found matching 'tests'
Adding pip 8.1.2 to easy-install.pth file
Installing pip script to /usr/local/bin
Installing pip2.7 script to /usr/local/bin
Installing pip2 script to /usr/local/bin

Installed /Library/Python/2.7/site-packages/pip-8.1.2-py2.7.egg
Processing dependencies for pip
Finished processing dependencies for pip
$ pip --version
pip 8.1.2 from /Library/Python/2.7/site-packages/pip-8.1.2-py2.7.egg
(python 2.7)
$
```

`pip` ツールが使用できるようになり、追加パッケージのインストールに利用できます。

Linux での PIP

Linux ディストリビューションでは、パッケージ マネージャを使用して `pip` パッケージをインストールします。たとえば、Ubuntu のパッケージは、`python-pip` になります。

```
$ apt-cache search python-pip
python-pip - alternative Python package installer
```

検証（すべてのプラットフォーム）

`pip` がインストールされているかを確認するには、グローバルな Python 環境で使用できるパッケージを参照します。

1. コマンドライン ウィンドウを開きます。
2. 「`pip list`」と入力します。

通常見込まれる出力は、次の例のようになります。

```
$ pip list
altgraph (0.10.2)
bdist-mpkg (0.5.0)
bonjour-py (0.3)
macholib (1.5.1)
matplotlib (1.3.1)
modulegraph (0.10.4)
numpy (1.8.0rc1)
pip (8.1.2)
py2app (0.7.3)
pyobjc-core (2.5.1)
...
pyobjc-framework-WebKit (2.5.1)
pyOpenSSL (0.13.1)
pyparsing (2.0.1)
python-dateutil (1.5)
pytz (2013.7)
scipy (0.13.0b1)
setuptools (1.1.6)
six (1.4.1)
xattr (0.6.4)
zope.interface (4.1.1)
$
```

上記の出力例は、Mac OS X にインストールされた PIP の出力を示しています。

次の例では、Windows で新しくインストールされた Python のインストール パッケージを示しています。

```
C:\Users\userid\Code>pip list
pip (8.1.1)
setuptools (20.10.1)
You are using pip version 8.1.1, however version 8.1.2 is available.
```

```
You should consider upgrading via the 'python -m pip install --upgrade pip' c
ommand.


C:\Users\userid\Code>
```

この場合、Windows により `pip` の新しいバージョンへのアップグレードが推奨されます。これは、コマンド ライン ウィンドウで以下のコマンドを入力することで実施できます。

`pip` をインストールすることで、パッケージの検索、インストール、アンインストール、アップグレードなどが行えるようになります。この機能は、このラボのモジュールの次のセクションで使用する `virtualenv` のパッケージをインストールする際に必要になります。

```
$ pip search virtualenv | grep '^virtualenv ('
virtualenv (15.0.3)       - Virtual Python Environment builder
$
```

仮想環境

`pip` をインストールすることで、グローバルな Python 環境に重要なパッケージの 1 つをインストールすることができるようになります。これは任意ですが、グローバルな Python 環境を*クリーンな環境*に維持する上でのベスト プラクティスと考えられます。これを実現するために、Python のパッケージ `virtualenv` を使用しています。これは、Python の仮想環境ビルダーです。

仮想環境は、Python の仮想環境を構築して、それぞれ別の場所の複数のプロジェクトで必要とされる依存関係を維持するためのツールとなります。
詳細については、こちらを参照ください。

仮想環境を有効にした場合、この Python 環境内のいくつかのパッケージのみが使用できます。追加で必要とするものがある場合は、個別にインストールする必要があります。これにより、グローバルな Python 環境に影響を与えることなく、ディレクトリ内のローカルもしくは特別な Python 環境を維持できます。

また、*nix 系のプラットフォームで管理者権限を使用して `pip` を実行する必要がなくなります。Python パッケージを、通常のユーザ権限でインストールすることができ、`sudo` コマンドが不要となります。これがもう 1 つの利点です。

次の手順で、`virtualenv` のインストール方法を示します。仮想環境の作成方法、有効化の方法、また、このような有効化された仮想環境に追加のパッケージをインストールする方法についても説明します。また後ほど、`requests` のパッケージが必要になります。これは、手順を示す際に使用されます。

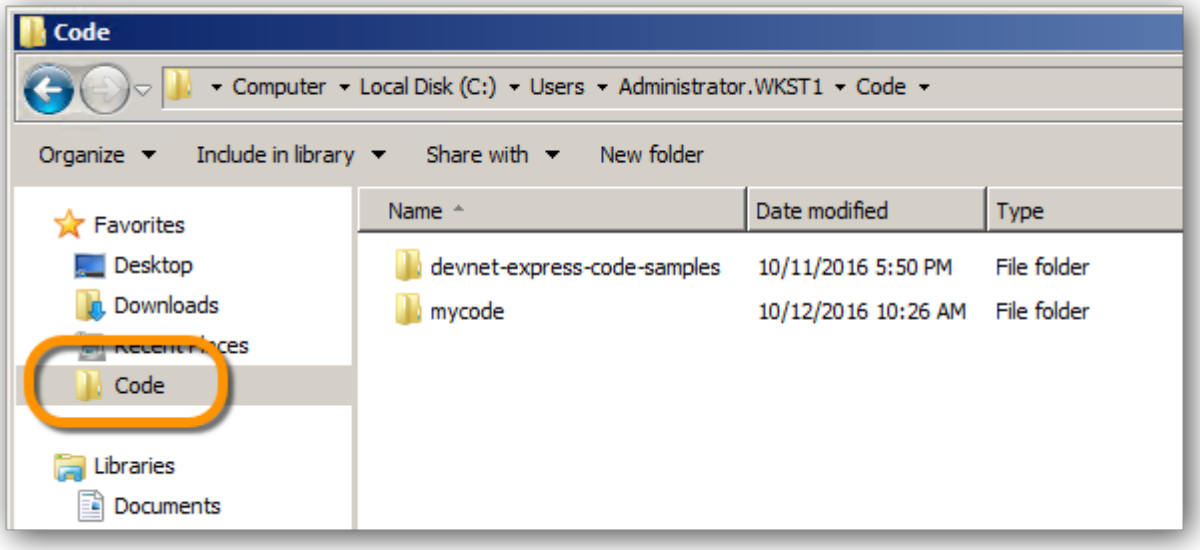
コード ディレクトリ

すべてのサンプル コード、仮想環境、自身の編集中のコードを保存するためのディレクトリを作成します。dCloud Windows ワークステーション上で、現在のユーザのホーム ディレクトリに「`Code`」という名称のディレクトリを用意します。

次の例では、ディレクトリ「`Code`」が作成され、現在のディレクトリであると仮定します。

dCloud ポッド内では、「`Code`」ディレクトリは次のような状態となります。

- Windows Explorer の *[お気に入り(Favorite)]* リンクが、dCloud Windows ワークステーションの「`C:\Users\Administrator.WKST1\Code`」にリンクされます。



- dCloud Ubuntu ワークステーションでは、「`/home/cisco`」ディレクトリの配下に置かれます。

Mac/Linux での仮想環境

- `pip` を使用して `virtualenv` をインストールします(これは `virtualenv` をインストールしていない場合に、一度だけ必要になります)。

```
$ sudo -H pip install virtualenv
Collecting virtualenv
  Downloading virtualenv-15.0.3-py2.py3-none-any.whl (3.5MB)
    100% |████████████████████████████████| 3.5MB 335kB/s
Installing collected packages: virtualenv
Successfully installed virtualenv-15.0.3
$
```

注:これは、管理者権限(`sudo`)で実行する必要があります。パッケージをグローバルにインストールするためです。また、「`-H`」の使用が推奨されます。root ユーザのホーム ディレクトリを正しく設定するためです。

- 必要な仮想環境ごとに、選択したディレクトリに設定する必要があります(ここで使用するディレクトリ/プロジェクト名は「`mycode`」です)。

```
$ virtualenv mycode
New python executable in mycode/bin/python
Installing setuptools, pip...done.
$
```

- 仮想環境を有効化します。

```
$ cd mycode
$ source bin/activate
(mycode)$
```

- 仮想環境に requests のパッケージをインストールします。

```
$ pip search requests | grep '^requests ('
requests (2.11.0)       - Python HTTP for Humans.
$ (mycode) $ pip install requests

Collecting requests
  Downloading requests-2.11.0-py2.py3-none-any.whl (514kB)
    100% |████████████████████████████████| 522kB 1.4MB/s
Installing collected packages: requests
Successfully installed requests-2.11.0
(mycode) $
```

- 仮想環境に追加のパッケージをインストールします。

```
(mycode) $ pip install netaddr argparse pyang
Collecting netaddr
```

```
  Using cached netaddr-0.7.18-py2.py3-none-any.whl
Collecting argparse
  Using cached argparse-1.4.0-py2.py3-none-any.whl
Collecting pyang
  Using cached pyang-1.7-py2.py3-none-any.whl
Collecting uniq
  Using cached uniq-1.2.1.28-py3-none-any.whl
Installing collected packages: netaddr, argparse, pyang, uniq
Successfully installed argparse-1.4.0 netaddr-0.7.18 pyang-1.7 uniq-1.2.1.28
(mycode) $
```

- 仮想環境に ncclient パッケージをインストールします（これは、NETCONF を使用するラボで必要になります）。

```
(mycode) $ pip install ncclient
.
.
.
Successfully installed ncclient paramiko lxml six cryptography pyasn1 idna cffi pycparser
Cleaning up...

(mycode) $
```

Windows での仮想環境

Windows OS で、仮想環境をインストールし、「`Scripts`」ディレクトリを探します。通常、仮想環境を有効化する際には、「`Scripts\activate`」を実行します。

同様に、追加のツールとコマンドは「`Scripts`」ディレクトリに保存されています。パスにディレクトリを追加するか、仮想環境のディレクトリから「`python Scripts\sometool`」を使用してコマンドを実行します。

- `pip` を使用して `virtualenv` をインストールします（これは `virtualenv` をインストールしていない場合に、一度だけ必要になります）。

```
C:\Users\userid\Code>pip install virtualenv
Collecting virtualenv
  Downloading virtualenv-15.0.3-py2.py3-none-any.whl (3.5MB)
    100% |################################| 3.5MB 273kB/s
Installing collected packages: virtualenv
Successfully installed virtualenv-15.0.3
You are using pip version 8.1.1, however version 8.1.2 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\Users\userid\Code>virtualenv --version
15.0.3
```

- 「mycode」という名称の仮想環境を作成します。

```
C:\Users\userid\Code>virtualenv mycode
Using base prefix 'c:\\python35'
New python executable in C:\Users\userid\mycode\Scripts\python.exe
Installing setuptools, pip, wheel...done.

C:\Users\userid\Code>
```

- 仮想環境を有効化します。

```
C:\Users\userid\Code>cd mycode

C:\Users\userid\Code\mycode>Scripts\activate.bat

(mycode) C:\Users\userid\Code\mycode>pip list

pip (8.1.2)
setuptools (25.1.6)
```

```
wheel (0.29.0)

(mycode) C:\Users\userid\Code\mycode>
```

注：有効化の方法は、使用中のシェルによって異なります。`cmd.exe` を使用している場合は、`activate.bat` を使用する必要があります。Cygwin や / もしくは Bash を使用している場合は、`activate`（Bash のスクリプト）を使用する必要があります。

プロンプトが変化することによって、それが動作したことが分かります。

- 仮想環境に追加のパッケージをインストールします。

```
(mycode) C:\Users\userid\Code\mycode>pip install requests netaddr argparse pyang uniq
Collecting requests
  Using cached requests-2.11.1-py2.py3-none-any.whl
Collecting netaddr
  Using cached netaddr-0.7.18-py2.py3-none-any.whl
Collecting argparse
  Using cached argparse-1.4.0-py2.py3-none-any.whl
Collecting pyang
  Using cached pyang-1.7-py2.py3-none-any.whl
Collecting uniq
  Using cached uniq-1.2.1.28-py3-none-any.whl
Installing collected packages: requests, netaddr, argparse, pyang, uniq
Successfully installed argparse-1.4.0 netaddr-0.7.18 pyang-1.7 requests-2.11.1 uniq-1.2.1.28

(mycode) C:\Users\userid\Code\mycode>
```

- 仮想環境に ncclient パッケージをインストールします（これは、NETCONF を使用するラボで必要になります）。

```
(mycode) C:\Users\userid\Code\mycode>pip install ncclient
.
.
```

- .
- `(mycode) C:\Users\userid\Code\mycode>`

> **注**:上記の手順だけで ncclient のインストールが完了することが理想です。しかし、実際には、コンパイラの欠如や UniCodeDecodeErrors に関するエラー メッセージが表示される場合があります。
>
> 次のセクションで Windows での ncclient のインストールについて確認してください。Windows での ncclient のインストールに問題が発生した際のインストール手順を示しています。

Python 3.5 を使用している Windows での ncclient のインストール

上述したとおり、Python 3.5 を使用している Windows での ncclient のインストールには問題があります。この文書の作成時点(2016 年 10 月)で、インストールを妨げる 2 つの大きな問題があります。

1. **lxml との依存関係**:ncclient には、lxml との依存関係があります。lxml を pip を使ってインストールするには、マシンで C 言語の開発環境が機能している必要があります。しかし、そのためには、Visual C やネット ランタイムおよび関連ライブラリなどのインストール作業が必要になります。また、これらのインストールに、合計で 4 GB 以上に相当するディスク容量が必要となります。

   この場合の比較的容易なアプローチは、こちらで説明するように http://www.lfd.uci.edu/~gohlke/pythonlibs/#lxml から wheel のバイナリをインストールすることです。
2. **UniCode のエンコーディング エラー。**pip インストールを使用して ncclient をインストールする場合、次のエラーが発生する場合があります。*UnicodeDecodeError: 'charmap' codec can't decode byte 0x90 in position 4336: character maps to <undefined>*

この場合、ソースからのインストールが役立ちます。

上記の問題を回避するには、下の手順にしたがって、ncclient をインストールします。

1. Python 環境用に lxml をダウンロードします。Python のバージョン(3.5)とアーキテクチャ(64 ビット)が一致する必要があります。確認するには、Python インストールを確認します。http://www.lfd.uci.edu/~gohlke/pythonlibs/#lxml から一致するバージョンをダウンロードします。ファイル名は、「`lxml-3.6.4-cp35-cp35m-win_amd64.whl`」(もしくは類似した名称)になります。重要な点は、ファイル名にプラットフォーム(「win_amd64」)と Python のバージョン(「cp35」)が含まれていることです。そうなっていない場合、pip は wheel のインストールを拒否します。
2. 有効となった仮想環境で、wheel をインストールします。

```
(mycode) C:\Users\userid\Code\mycode>pip install ..\Downloads\lxml-3.6.4-cp35-cp35m-win_amd64.whl
Processing c:\users\userid\downloads\lxml-3.6.4-cp35-cp35m-win_amd64.whl
Installing collected packages: lxml
Successfully installed lxml-3.6.4

(mycode) C:\Users\userid\Code\mycode>
```

3. GitHub から ncclient のソースを複製します。この作業は、現時点では、Python 3.5 を使用している場合にのみ必要です。Python 2.7 であれば、文字エンコードに関してはさらに容易になります。Python 2.7 を使用し、'pip install ncclient' でインストールしている場合は、これ以上の作業は必要ありません。

```
(mycode) C:\Users\userid\Code\mycode>git clone https://github.com/ncclient/ncclient.git
Cloning into 'ncclient'...
remote: Counting objects: 3734, done.
remote: Compressing objects: 100% (2/2), done.
Receivinremote: Total 3734 (delta 0), reused 0 (delta 0), pack-reused 3732
Receiving objects: 100% (3734/3734), 2.68 MiB | 0 bytes/s, done.
Resolving deltas: 100% (2086/2086), done.
Checking connectivity... done.

(mycode) C:\Users\userid\Code\mycode>
```

4. 作業ディレクトリを「ncclient」ディレクトリに変更します（cd コマンドを使用します）。

```
(mycode) C:\Users\userid\Code\mycode>cd ncclient
```

5. ncclient の requirements をインストールします。

```
(mycode) C:\Users\userid\mycode\ncclient>pip install -r requirements.txt
.
.
.
(mycode) C:\Users\userid\mycode\ncclient>
```

6. ncclient をインストールします。

```
(mycode) C:\Users\userid\mycode\ncclient>py setup.py install
```

上記の手順が正常に完了すると、ncclient パッケージとその依存関係がインストールされます。

パッケージ インストールの確認（すべてのプラットフォーム）

この時点で、「`mycode`」ディレクトリ内の Python インストールに、次のパッケージが含まれているはずです。Windows OS の場合、出力とプロンプトに少し相違の出る場合がありますが、パッケージは同一であることに留意してください。

```
(mycode) $ pip list
cffi (1.8.3)
cryptography (1.5.2)
enum34 (1.1.6)
idna (2.1)
ipaddress (1.0.17)
lxml (3.6.4)
ncclient (0.5.2)
netaddr (0.7.18)
paramiko (2.0.2)
pip (8.1.2)
pyang (1.7)
pyasn1 (0.1.9)
pycparser (2.14)
requests (2.11.1)
setuptools (28.3.0)
six (1.10.0)
uniq (1.2.1.28)
wheel (0.30.0a0)
(mycode) $
```

Python 環境が機能していることを確認するには、ターミナル ウィンドウで次のコマンドを実行して、これらのステップを実行します。

1. コマンドライン ウィンドウを開きます（ターミナル ウィンドウ）。
2. 作業ディレクトリを virtualenv があるディレクトリに変更します。たとえば、「mycode」という名称のディレクトリにインストールされている場合は、次のように入力します。

   `cd mycode`
3. 次のように入力して、virtualenv を有効化します。

   `source bin/activate`
4. 次のように入力して、対話形式の Python セッションを開きます。

   `python`
5. Python セッション内で、次のように入力して、インストール済みの Python パッケージをインポートします。

```
6. import ncclient, netaddr, pyang, requests, ipaddress, uniq
```

次の例は、上記の手順を実行した場合に期待される出力です。

```
$ cd mycode
$ source bin/activate
(mycode)$ python
Python 2.7.10 (default, Oct 23 2015, 19:19:21)
[GCC 4.2.1 Compatible Apple LLVM 7.0.0 (clang-700.0.59.5)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import ncclient, netaddr, pyang, requests, ipaddress
>>> exit()
(mycode) $
```

おめでとうございます。Python 環境が機能する状態になりました。この環境には、`pip` パッケージ マネージャとインターネット上の PyPI パッケージ レポジトリからインストールした、いくつかの追加パッケージが含まれます。また、「`virtualenv`」コマンドを使用して、カスタマイズされた仮想 Python 環境を構築および有効化することができます。

注：上記の出力のバージョンは、特定の Python バージョンを示しています（上記では、Mac OS X 10.11 にネイティブの Python 2.7 がインストールされています）。実際の Python バージョンのバナーは異なる場合があります。特に Python 3.5 をインストールした場合は異なる可能性があります。重視すべき項目は、`import` ステートメントです。ここに特にエラーが生じなければ、すべて正常です。

ローカル マシンの準備が完了し、Python でのプログラミングを続けることができます。

以上で終了です。

次のページに進み、シナリオ 1 ～ 4 に向けて、リモートの dCloud ラボ ポッドへの接続設定を実施します。dCloud が提供されないインフラストラクチャ コンポーネント（独自インフラストラクチャなど）を使用している場合は、次の 2 つのステップを省略し、ステップ 6「*アカウントのサインアップ*」に直接移動します。

自分のコンピュータを設定する方法
このラボでは導入部分を扱います。

ステップ 4:ラボ環境(dCloud ポッド)との接続設定

シナリオ 5(ラボ環境に自身のインフラストラクチャを使用する)を設定している場合、このステップを省略してステップ 6 に移動します。

次のシナリオを設定している場合は、このページのステップを実行する必要があります。

- シナリオ 1(dCloud の開発ツールを使用して、ハンズオン クラスに参加)
- シナリオ 2(ローカル マシンの開発ツールを使用して、ハンズオン クラスに参加)
- シナリオ 3(dCloud の開発ツールを使用して、自身の環境から作業)
- シナリオ 4(ローカル マシンの開発ツールを使用して、自身の環境から作業)

ブラウザ ベースの RDP セッションの使用

ローカル マシンの開発ツールを使用する場合は、このセクションを省略して、下の AnyConnect のインストールのセクションに進むことができます。

ハンズオン イベントで、Web ベースのリモート デスクトップを使用する場合は、次が必要になります。

- HTML5 をサポートするブラウザ、もしくは AnyConnect のインストール
- CCO アカウント

ご注意ください:リモート デスクトップを使用する場合は、ローカル マシンにファイルを保存できません。リモート デスクトップを使用する場合は、Dropbox や Box または類似のファイル共有サービスのアカウントを用意することを推奨します。これは、ラボの演習中に作成したファイルを保存できるようにするためです。たとえば、イベントの終了後に、作成したコード ファイルを保存することができます。

ブラウザベースの RDP 接続を使用してリモート ラボ環境にアクセスする場合は、ハンズオン イベントの最中に URL が提供されます。必要な作業は、ブラウザにこの URL をコピー アンド ペーストすることだけです。これでデスクトップ セッションがブラウザにロードされます。

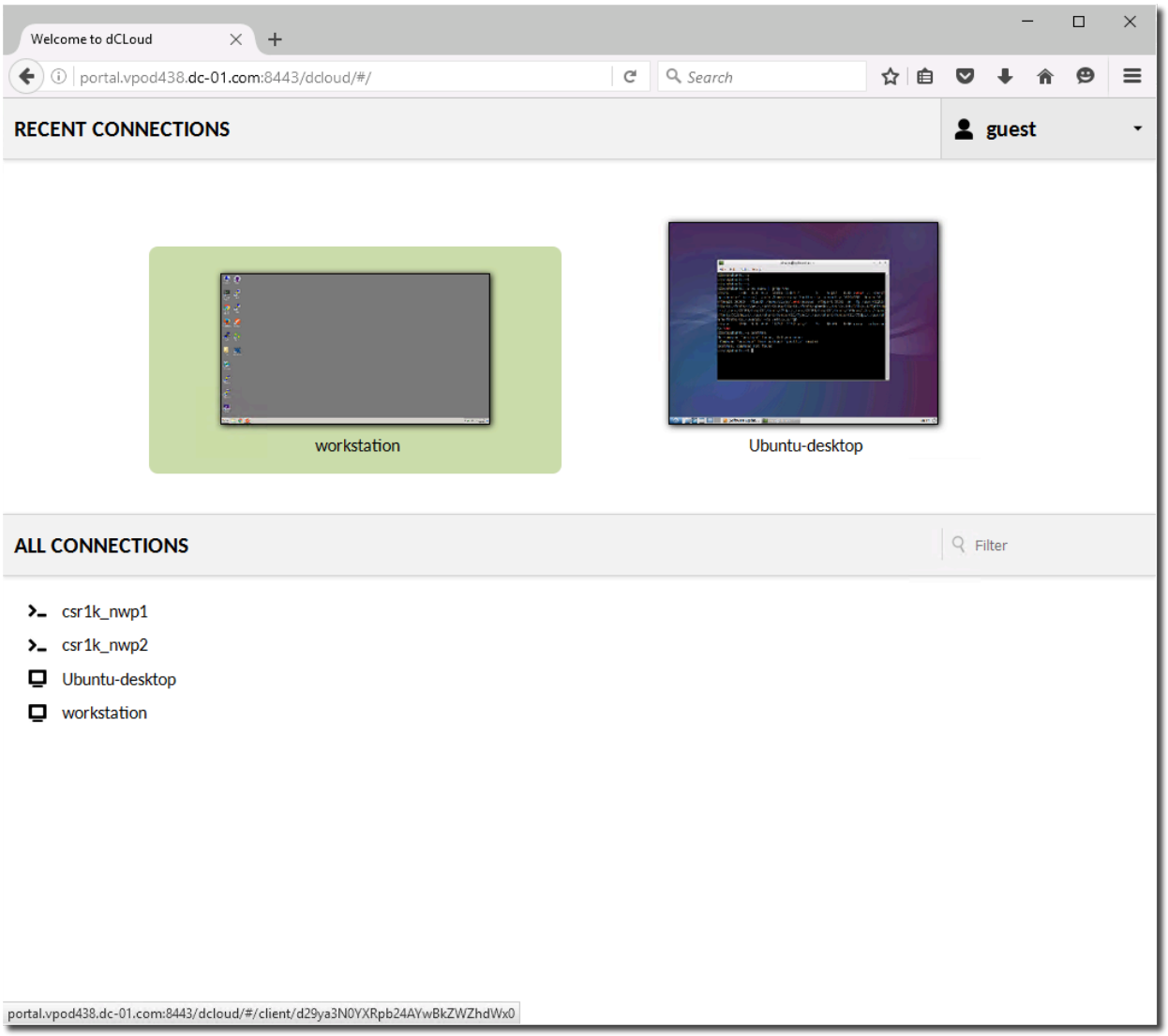
URL は次のようなものになります。

```
http://vXXXuser1:XXXXXX@portal.vpodXXX.dc-01.com:8443/dcloud
```

独自の dCloud 環境をプロビジョニングしている場合(また、リモート ラボが提供されるイベントに参加していない場合)は、dCloud のダッシュボードで URL を確認する必要があります。

- 付録に記載されているように、dCloud のダッシュボードからユーザ名、パスワード、DNS アドレス/パブリック アドレスを取得します。
- それらの詳細を、URL の適切な場所に入力します(上記のサンプル URL の XXX の箇所)。
- 完成した URL をブラウザに貼り付けます。

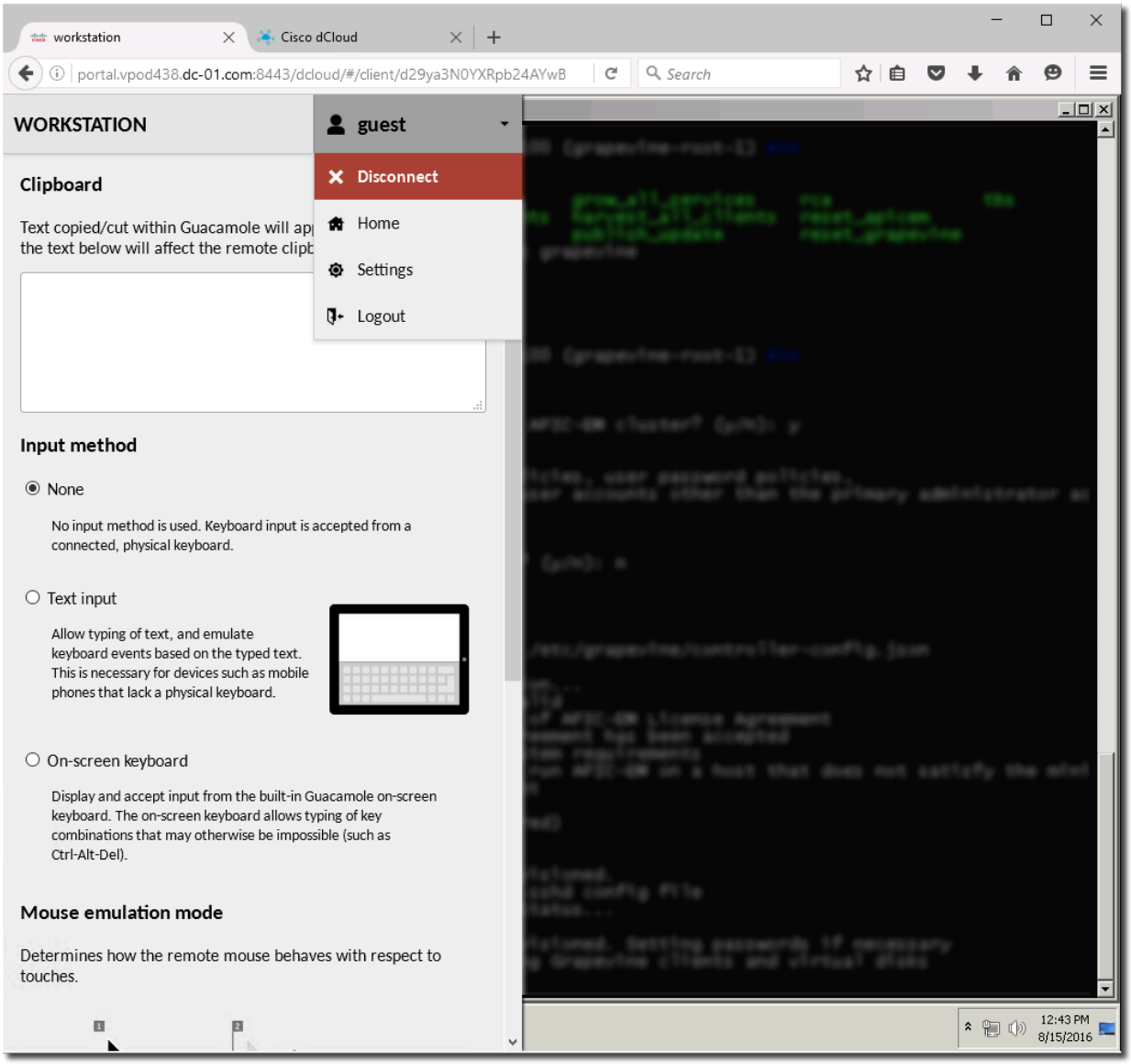
ブラウザの RDP セッションの接続画面が表示されます。



接続するデバイスを選択します。Windows ワークステーションにアクセスすると、画面全体を利用できるように、ブラウザの「全画面モード」を有効にすることを推奨されます。全画面モードに切り替えた後、デスクトップの左右いずれかの黒い領域を右クリックして、[リロード（Reload）] アイコンを選択します。これで、Windows が画面のネイティブ解像度を使用できるようになります。

Ubuntu のデスクトップでは、画面解像度の動的変更はサポートされておらず、1024 X 768 の解像度に固定されています。

注：Ctrl+Shift+Cmd（Mac キーボード）、もしくは Ctrl+Shift+Win（Windows キーボード）で、共有クリップボードや追加設定にアクセスできます。



AnyConnect の設定（VPN アクセス用）

注:リモート ラボ環境の一部であるリモート デスクトップ セッションを使用する場合は、「AnyConnect」のセクションは省略できます。このケースで必要となるのは、Chrome、Firefox などの HTML5 対応ブラウザのみです。
ローカル マシンの開発ツールを使用する場合は、dCloud ポッドのラボ環境へのアクセスに VPN を使用する必要があります。

このセクションでは、AnyConnect のインストール方法について説明します。すでにインストールされている場合は、次のセクションへ移動して、接続のテストを行ってください。

インストールされていない場合、AnyConnect は次の 3 つの方法で入手することができます。

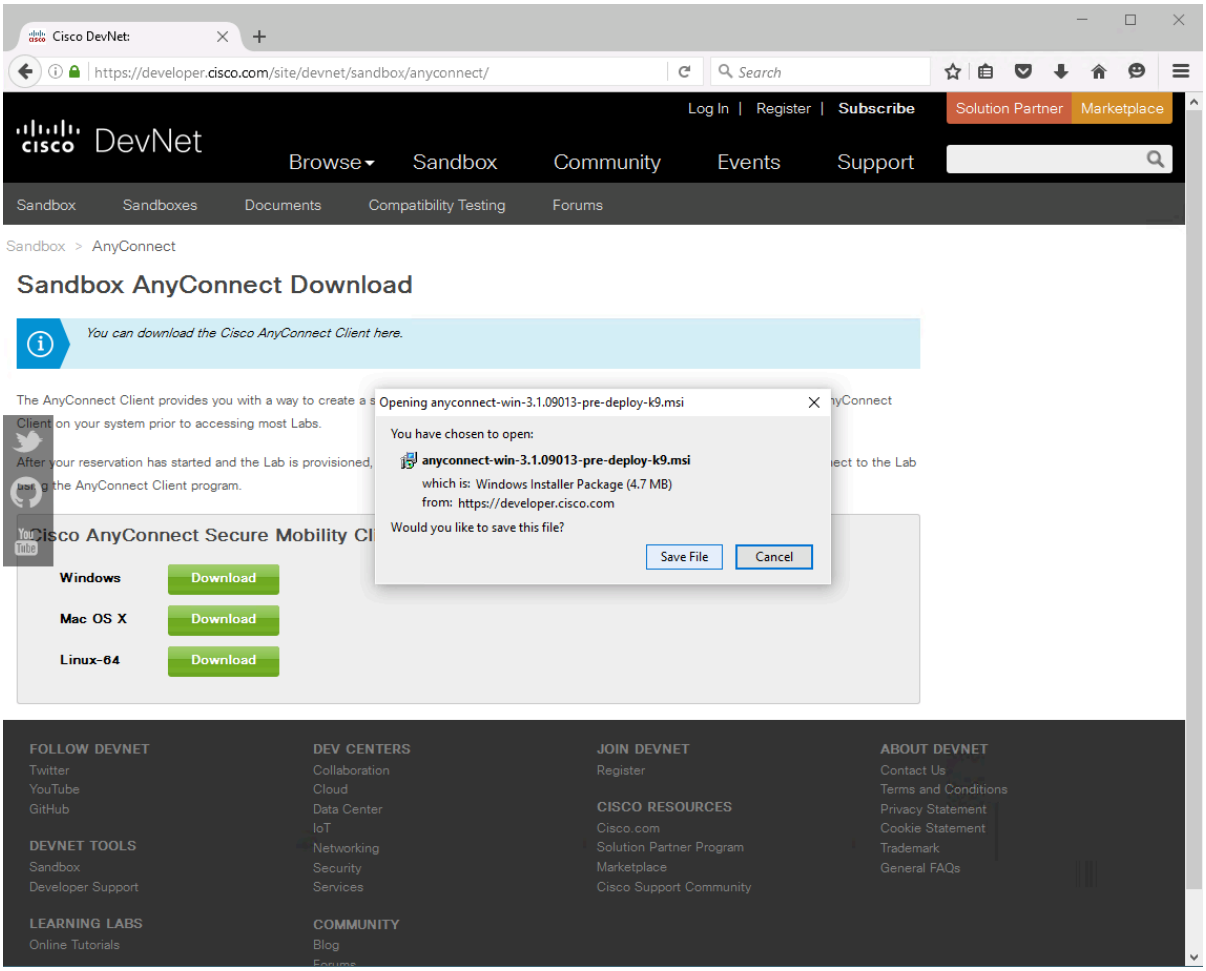
1. http://www.cisco.com/cisco/web/portal/support/products/home.html?cid=283000185&locale=ja_JP から。シスコからソフトウェアをダウンロードする際に、CCO アカウントが必要になります。バージョン 3 または 4 を選択できますが、どちらでもかまいません。
2. Cisco DevNet サンドボックスから。サンドボックスでは、Windows、Mac、Linux 用のクライアント（バージョン 3.x）が提供されています。
3. VPN を提供する ASA から直接。dCloud および DevNet サンドボックスの両方で、接続中にダウンロード用のクライアントが提供されます。初回にクレデンシャルが必要になります。

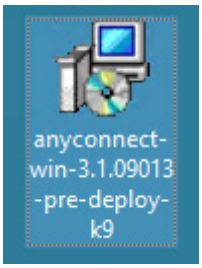
ヒント：AnyConnect のダウンロード方法としては、オプション 2 が最も簡単な方法として推奨されます。

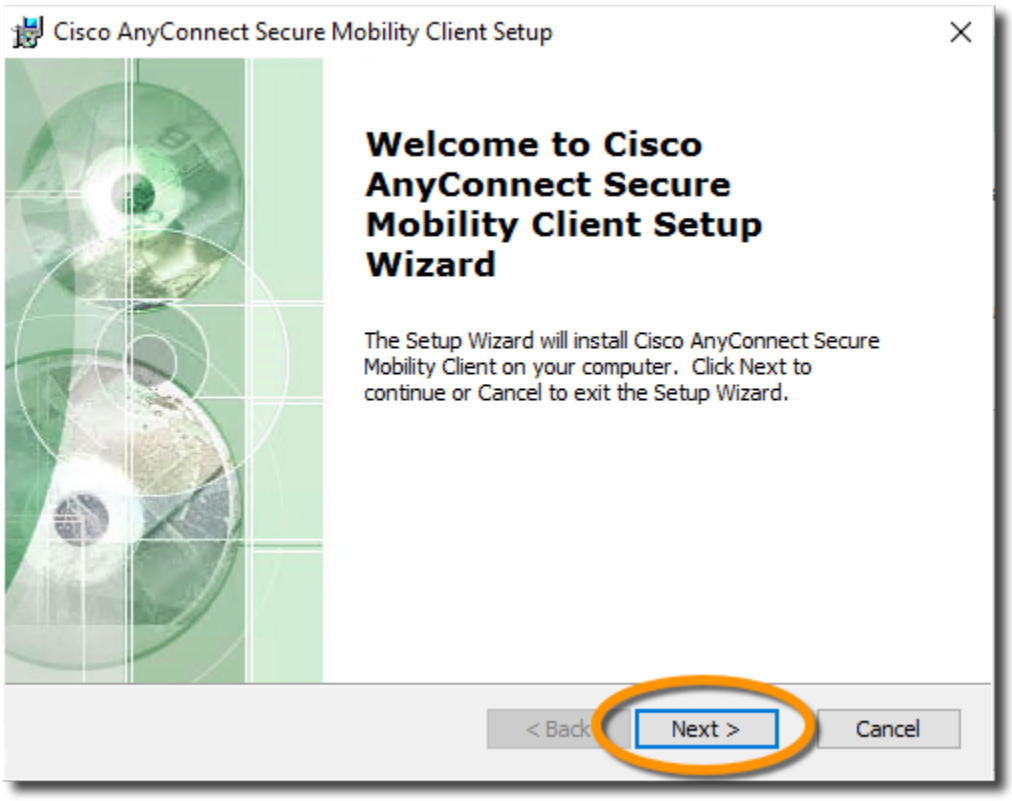
AnyConnect のインストール

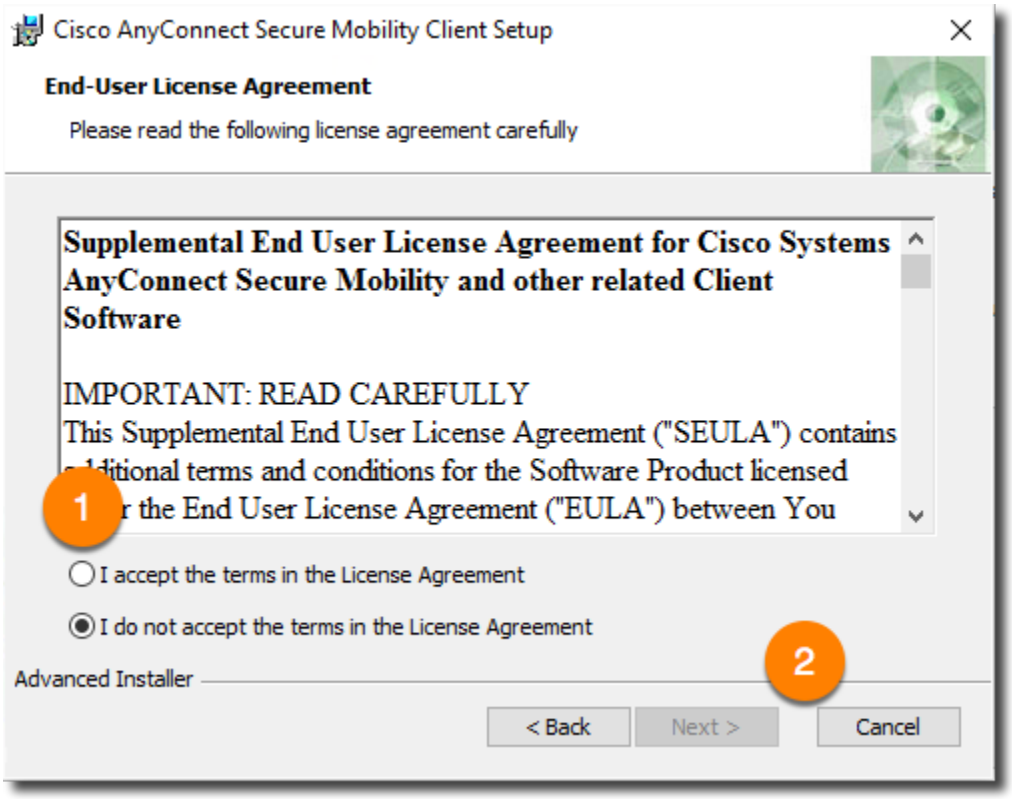
64 ビットの Windows、Mac、Linux

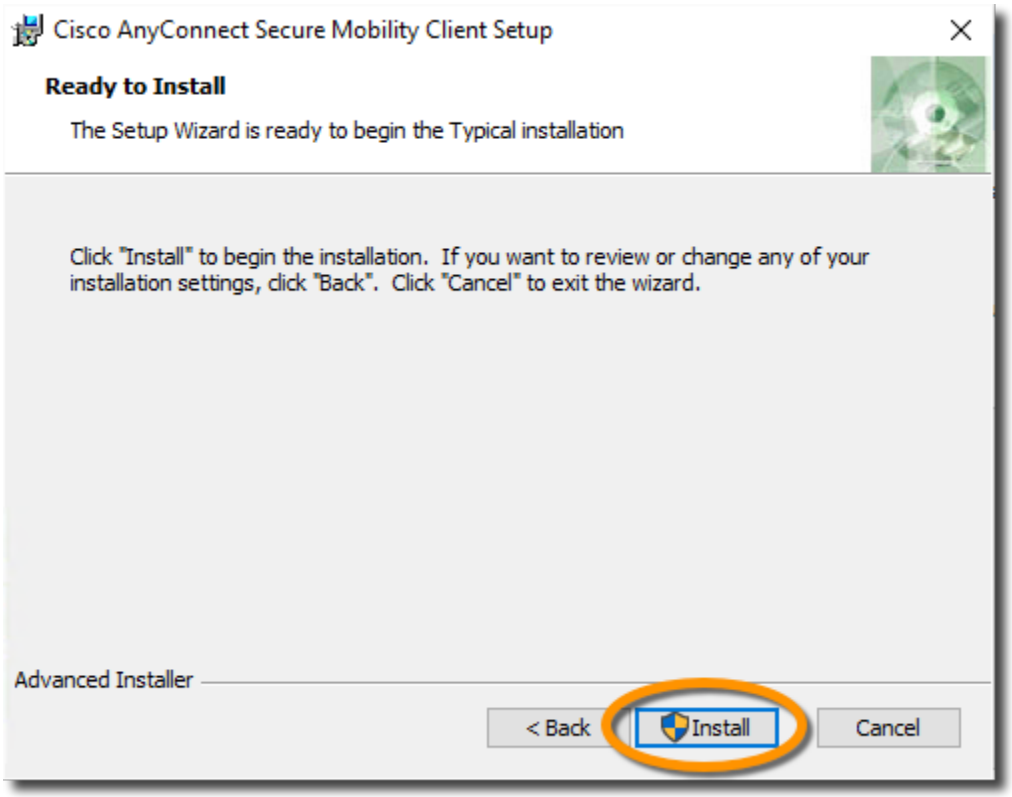
直前のセクションで示されたソースから適切なインストール パッケージをダウンロードして、PC に保存します（Windows のインストールを次に表示）。

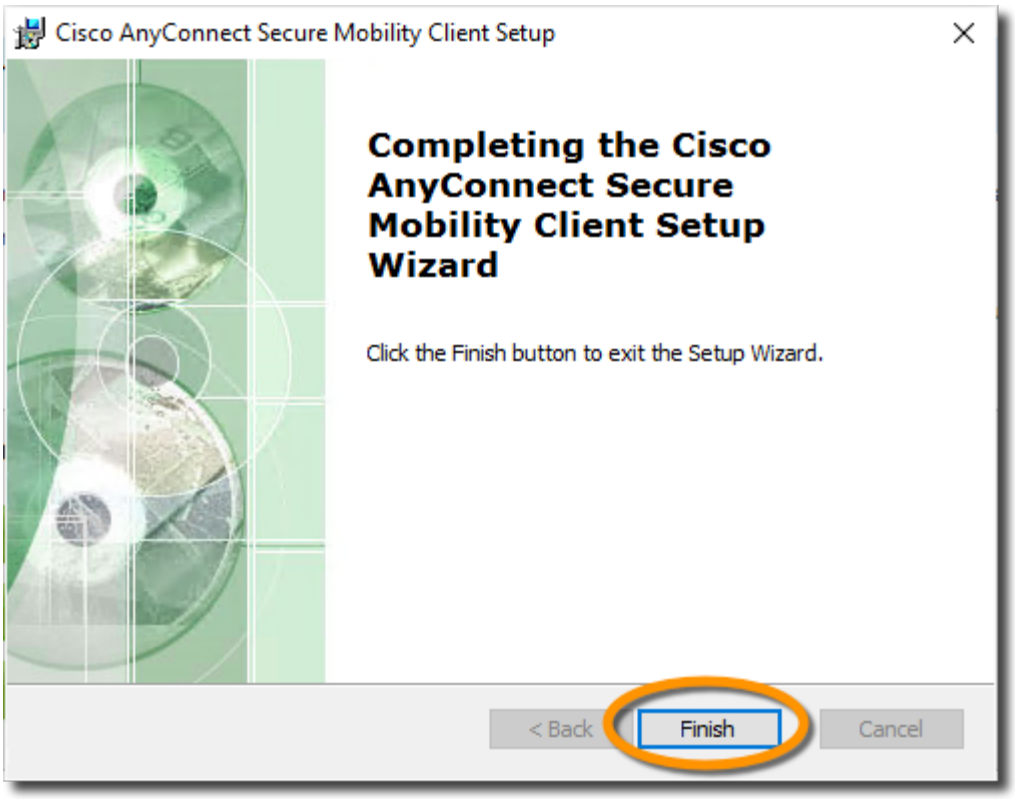



anyconnect-
win-3.1.09013
-pre-deploy-
k9


Cisco AnyConnect Secure Mobility Client Setup
Welcome to Cisco AnyConnect Secure Mobility Client Setup Wizard
The Setup Wizard will install Cisco AnyConnect Secure Mobility Client on your computer. Click Next to continue or Cancel to exit the Setup Wizard.
< Back
Next >
Cancel


Cisco AnyConnect Secure Mobility Client Setup
End-User License Agreement
Please read the following license agreement carefully
Supplemental End User License Agreement for Cisco Systems AnyConnect Secure Mobility and other related Client Software
IMPORTANT: READ CAREFULLY
This Supplemental End User License Agreement ("SEULA") contains
r the End User License Agreement ("EULA") between You
1
I accept the terms in the License Agreement
I do not accept the terms in the License Agreement
2
Advanced Installer
< Back
Next >
Cancel


Cisco AnyConnect Secure Mobility Client Setup
Ready to Install
The Setup Wizard is ready to begin the Typical installation
Click "Install" to begin the installation. If you want to review or change any of your installation settings, click "Back". Click "Cancel" to exit the wizard.
Advanced Installer
< Back
Install
Cancel



Mac OS X バージョンのインストール手順は、Windows のインストールとほぼ同じです。インストール中に、クレデンシャルを入力する必要があります。

Linux バージョンの場合は若干異なります（自己解凍方式のシェル アーカイブを使用します）が、Linux の使用方法に習熟していれば方法は理解できるはずです。

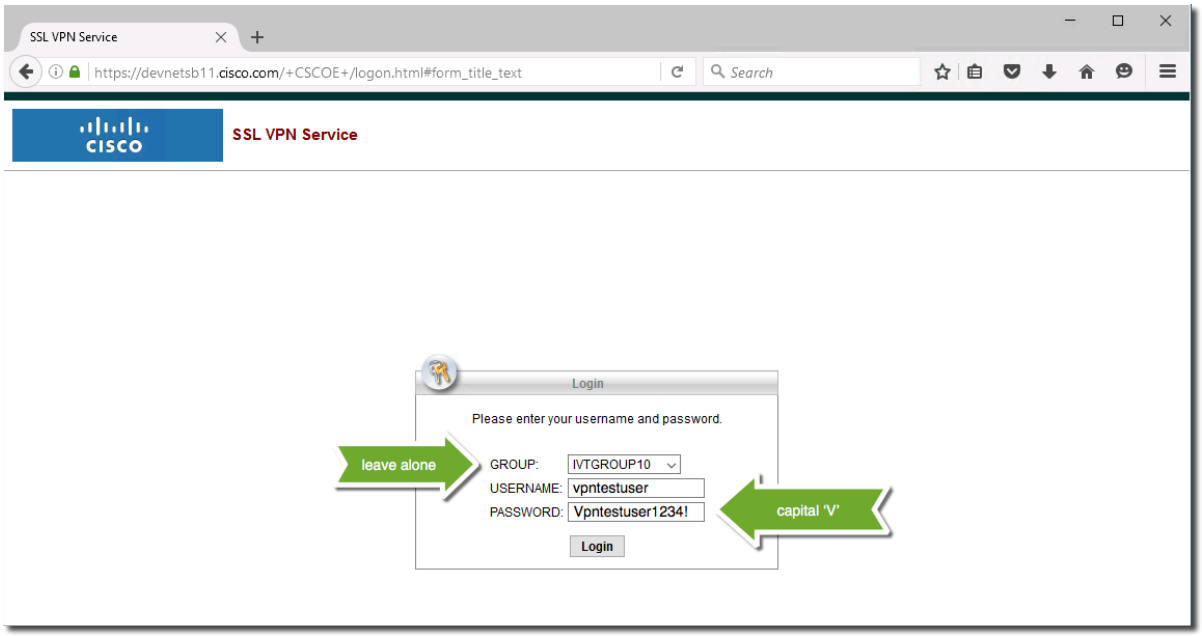
VPN ゲートウェイからの直接ダウンロード

重要な一般注意事項：サンドボックスの AnyConnect ソフトウェア パッケージは両方の環境で機能する必要があることから、このアプローチの使用を推奨します。ゲートウェイからのダウンロード、または、AnyConnect の章の最初にあるダウンロード ページを利用してください。

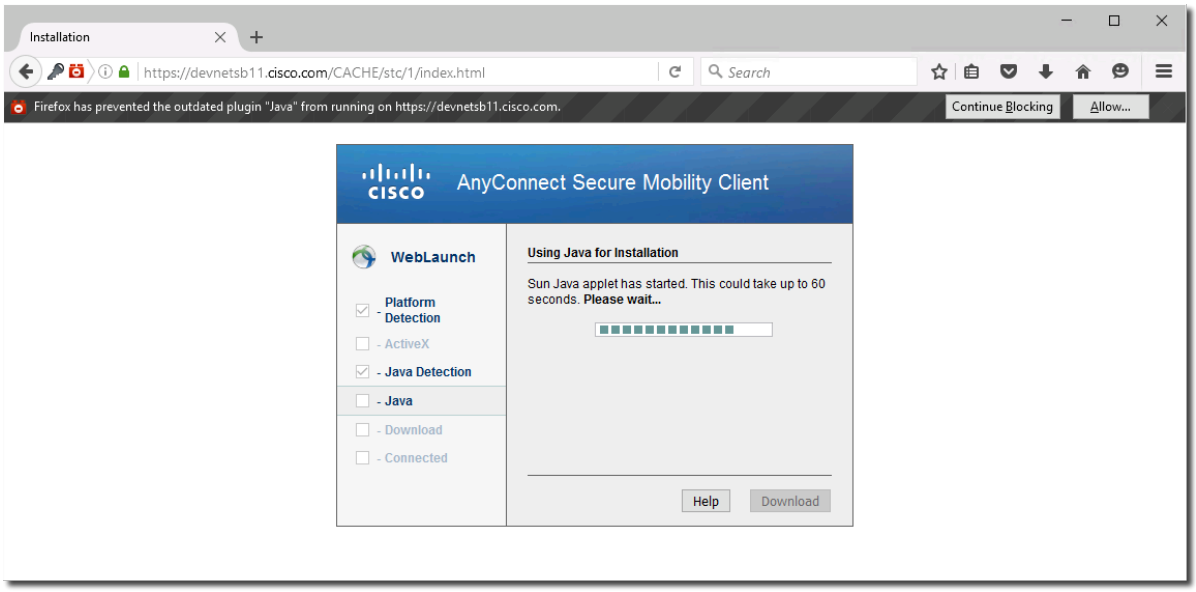
また、VPN ゲートウェイ自体をブラウザで指定することもできます。ただ、この作業を行うには、クレデンシャルを入力する必要があります。イベントの際には、ログイン情報シートで提供されるクレデンシャルを使用します。dCloud でセルフ プロビジョニングされたラボ ポッドを実行している場合は、セッションの [セッションの詳細（Session Details）] ページ上の使用可能なクレデンシャルを使用します。下の*「dCloud VPN クレデンシャルの取得」*のセクションを参照してください。

スケジュールされたセッションを確認するには、適切な dCloud DC にログインする必要があります。VPN のヘッドエンド リンクは、サイトによって異なります。これらは常に「`https://dcloud-XXX-anyconnect.cisco.com/`」の形式になります（XXX はサイト ロケータ）。「`rtp`」、「`lon`」、「`sng`」があり、それぞれローリー（米国ノースカロライナ州）、ロンドン（英国、EMEAR）、シンガポール（APAC）を意味します。

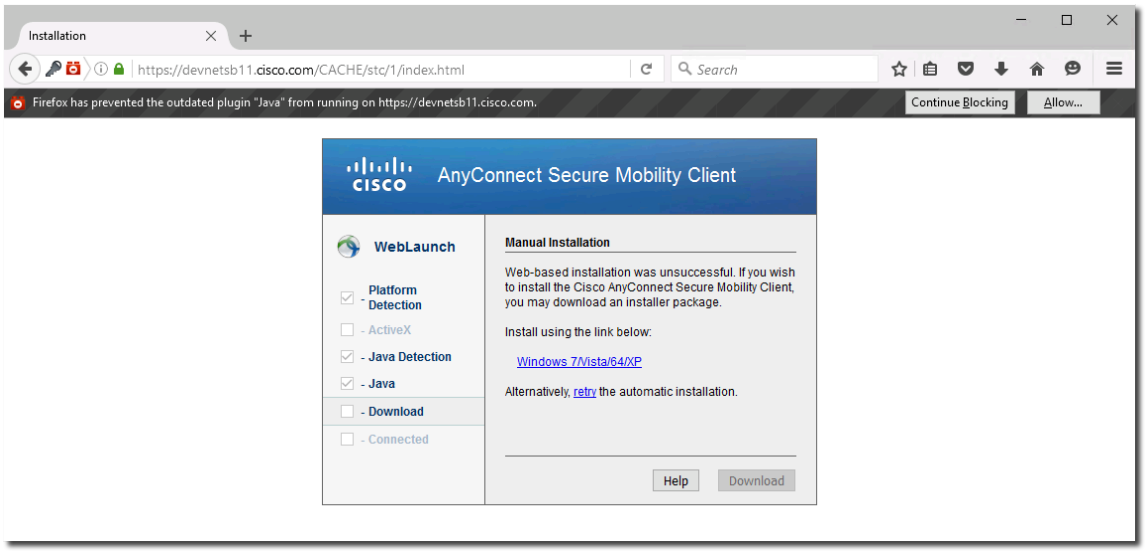
1. 適切なヘッドエンドを、ブラウザで指定します。それぞれの URL は次のようになります。
    - サンドボックス：https://devnetsb11.cisco.com/
      ユーザ名「`vpntestuser`」、パスワード「`Vpntestuser1234!`」
    - dCloud アメリカ：https://dcloud-rtp-anyconnect.cisco.com/
    - dCloud EMEAR：https://dcloud-lon-anyconnect.cisco.com/
    - dCloud APAC：https://dcloud-sng-anyconnect.cisco.com/
2. クレデンシャルを入力します。dCloud VPN ゲートウェイのユーザ名とパスワードを取得する方法については、下の*「dCloud VPN クレデンシャルの取得」*のセクションを参照ください。



3. ログインすると、ブラウザ アプリケーションは AnyConnect のパッケージをダウンロードしようとします。



4. 最終的に、この画面が表示され、使用しているプラットフォームの一致する AnyConnect のパッケージをクリックしてダウンロードすることができます（例では Windows）。



OpenConnect

OpenConnect は、代替となるオープン ソースの無料の AnyConnect VPN クライアントです。dCloud とサンドボックス環境で機能します。これは、AnyConnect のネイティブ インストールが利用できない Linux やその他のプラットフォーム（BSD 系など）の場合に最適です。ほとんどの Linux ディストリビューションで、コンパイル前のバイナリ パッケージが利用可能です。

- Ubuntu：

```
ubuntu:~$ apt-cache search openconnect
[...]
openconnect - open client for Cisco AnyConnect VPN
vpnc-scripts - Network configuration scripts for VPNC and OpenConnect
ubuntu:~$
```

- RPM ベース：

```
[centos~]$ yum search openconnect
Loaded plugins: fastestmirror
Determining fastest mirrors
 * base: ftp.usf.edu
 * epel: mirror.math.princeton.edu
 * extras: ftp.usf.edu
 * rpmforge: ftp.nluug.nl
```

```
 * updates: ftp.usf.edu
 [...]
 openconnect.i686 : Open client for Cisco AnyConnect VPN
 openconnect.x86_64 : Open client for Cisco AnyConnect VPN

 Name and summary matches only, use "search all" for everything.
 [centos~]$
```

以下は、dCloud の VPN ゲートウェイとの接続を確立する際に使用できるスクリプトです。スクリプト冒頭の変数は、ユーザ番号やパスワードに置き換える必要があります（dCloud 内のすべての VPN ユーザは、vXXXuser1、vXXXuser2、vXXXuser16 という名前になります。XXX は番号です）。

```
#!/bin/bash
USER="535"
PASS="879c9d"
NUMB="1"
DC="rtp"
echo $PASS | openconnect \
    --user=v${USER}user${NUMB} \
    --passwd-on-stdin \
    https://dcloud-${DC}-anyconnect.cisco.com
```

サンドボックスの接続テストなどに、スクリプトなしでセッションを確立するには、次のように OpenConnect をコールします。

```
$ sudo openconnect https://devnetsb11.cisco.com/
Password:
POST https://devnetsb11.cisco.com/
Connected to 64.103.26.52:443
SSL negotiation with devnetsb11.cisco.com
Connected to HTTPS on devnetsb11.cisco.com
Got HTTP response: HTTP/1.0 302 Object Moved
GET https://devnetsb11.cisco.com/
Connected to 64.103.26.52:443
SSL negotiation with devnetsb11.cisco.com
Connected to HTTPS on devnetsb11.cisco.com
```

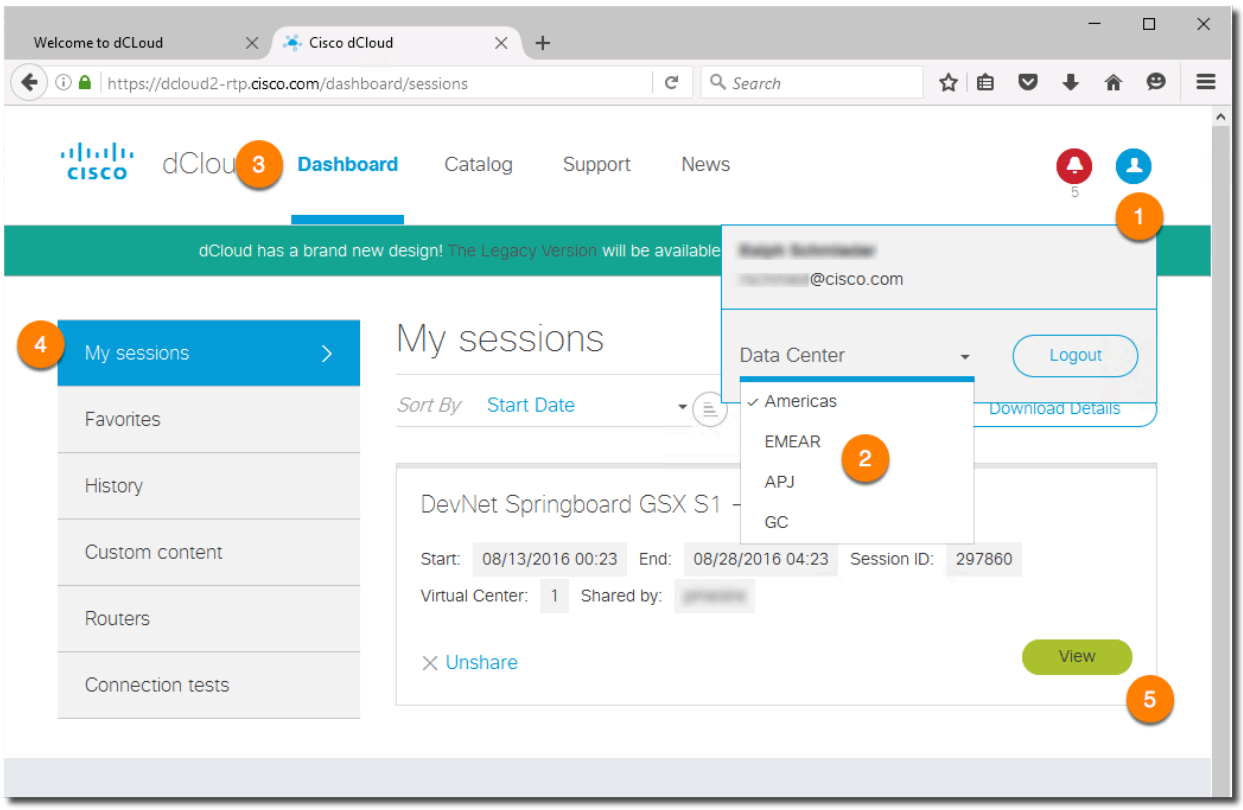
```
Got HTTP response: HTTP/1.0 302 Object Moved
GET https://devnetsb11.cisco.com/+webvpn+/index.html
SSL negotiation with devnetsb11.cisco.com
Connected to HTTPS on devnetsb11.cisco.com
Please enter your username and password.
GROUP: [IVTGROUP10|IVTGROUP100]:IVTGROUP10
Please enter your username and password.
Username:vpntestuser
Password:
POST https://devnetsb11.cisco.com/+webvpn+/index.html
Got CONNECT response: HTTP/1.1 200 OK
CSTP connected.DPD 30, Keepalive 20
Set up DTLS failed; using SSL instead
Connected as 172.16.255.24, using SSL
add host 64.103.26.52: gateway 172.16.33.1
add net 172.16.255.0: gateway 172.16.255.24
route: writing to routing socket: File exists
add net 172.16.1.0: gateway 172.16.255.24: File exists
route: writing to routing socket: File exists
add net 172.16.255.0: gateway 172.16.255.24: File exists
add net 10.10.0.0: gateway 172.16.255.24
add net 10.10.10.1: gateway 172.16.255.24
```

OpenConnect は自動的にバックグラウンドでの動作には移行**しません**。Ctrl+C を押して OpenConnect を停止できます。

dCloud のクレデンシャルの取得

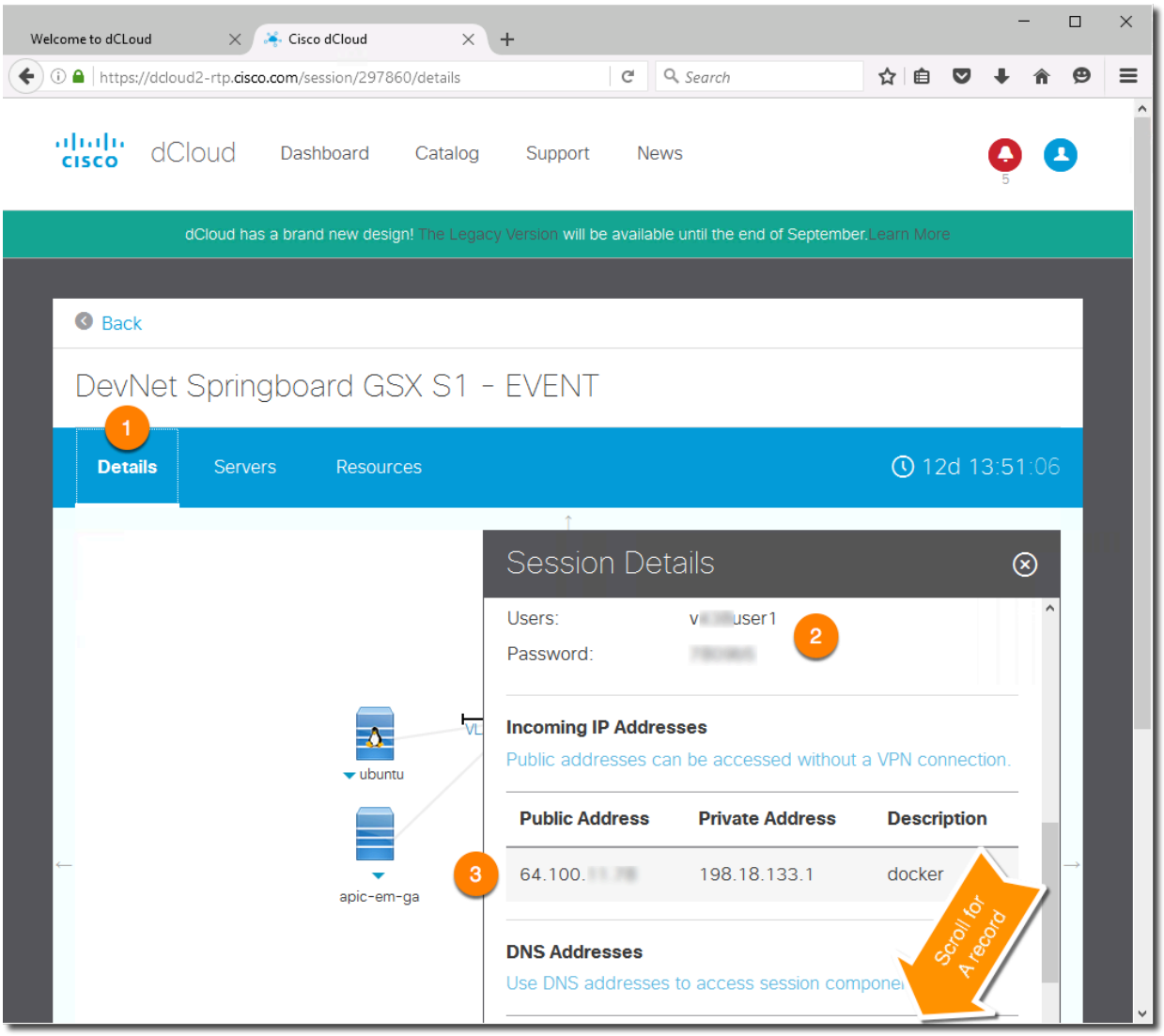
dCloud のセッション詳細にアクセスするには、次の手順に従います。

- https://dcloud.cisco.com に移動し、CCO クレデンシャルでログインします
- ラボ ポッドを実行している DC が正しく選択されていることを確認します（アメリカ地域、EMEAR、APJ、GC）（図の 1 および 2 で示されます）
- [ダッシュボード（Dashboard）]（図の 3）をクリックします
- [マイ セッション（My Sessions）]（図の 4）を選択します
- [表示（View）]（図の 5）をクリックします



さらに、[詳細（Details）]（図の 1）を選択して、AnyConnect クレデンシャルにアクセスします。以下をメモに書き留めます。

- ユーザ名およびパスワード（図の 2）
- パブリック IP アドレスおよびそれに関連する DNS レコード（スクリーン ショットには表示されていません。下まで画面をスクロールさせてください）（図の 3）
- AnyConnect のホスト URL（AnyConnect VPN を通して接続する場合）



AnyConnect VPN クライアントの設定、またはブラウザベースのリモート デスクトップ クライアント URL の作成に、上記の情報を使用します。

次の手順

シナリオ 1 ～ 4 に VPN 接続を使用する場合は、次のページに進んで接続のテストを行ってください。また、シナリオ 2 および 4 で、VPN と Python 環境の両方のテストを行うこともできます。

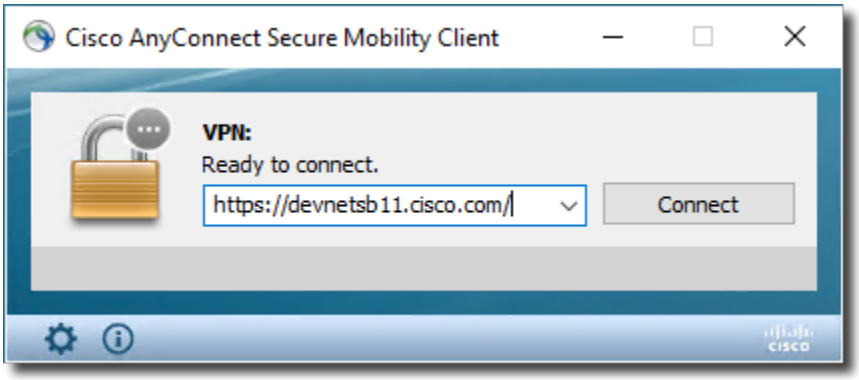
自分のコンピュータを設定する方法
このラボでは導入部分を扱います。

ステップ 5：ラボ環境（dCloud ポッド）への接続テスト

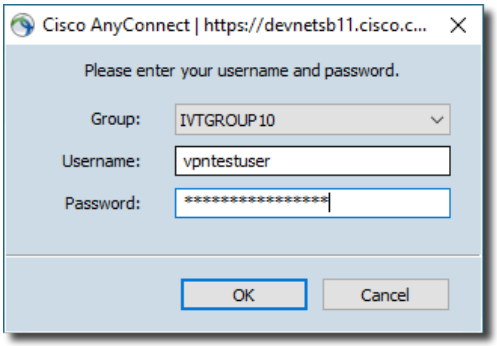
接続の確立

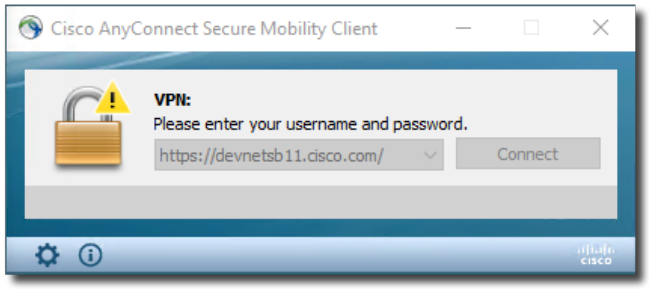
VPN クライアント（AnyConnect または OpenConnect）のインストールが正常に完了したら、次に接続のテストを行います。それでは始めましょう。

1. AnyConnect を開き、https://devnetsb11.cisco.com/ を入力します。[接続（Connect）] をクリックします。

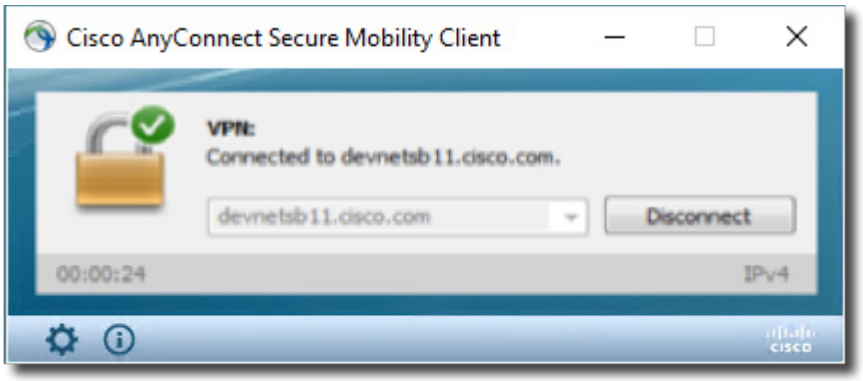


2. [IVTGROUP10] グループを選択します。通常はデフォルトで選択されています。
   ユーザ名：`vpntestuser`
   パスワード：`Vpntestuser1234!`（V は大文字です）





3. この時点で、接続が確立されます。



また、OpenConnect が使用されている場合は、OpenConnect 経由で VPN 接続することもできます。

テスト ページへの移動

接続が確立された後、モックアップ ラボ環境の VPN 内のコンポーネントに実際に到達できることを確認する必要があります。実際のハンズオン イベントで、マシンが接続できることを確認します。これは、スケジューリングの実施後、ラボ ポッドが起動するのを待つ間に、VPN の設定が機能していることを確認するのにも役立ちます。

任意のブラウザを開き、http://10.10.10.104 を指定します。このアドレスは、ラボの VPN 内であることに注意してください。ブラウザがプロキシを使用していないことを確認してください。プロキシを使用していると接続が阻害される場合があります。すべてが適切に設定されていれば、ブラウザは次のページを表示します。

ページが表示されたでしょうか。表示されていれば、設定が適切に完了しています。次の Python のインストール テストのセクションに進んでください。

上記のページが表示されない場合は、次のトラブルシューティングの手順を試してください。

- サーバへの Ping を実行します（`ping 10.10.10.104`）。動作するでしょうか。
- コンピュータのルーティング テーブルに、10 のネットワークのエントリが含まれていることを確認します。通常は、`netstat` コマンド（Mac/Linux）、または `route print` コマンド（Windows）を使用して実施します。

```
$ netstat -rn -finet | grep ^10
10.10/16           172.16.255.24      UGSc            2        2   utun0
10.10.10.1/32      172.16.255.24      UGSc            2        0   utun0
$
```

出力の最後にある `utun0` デバイスは、`10.10/16` プレフィックスへのトラフィックがトンネルを経由していることを示しています。

- プロキシ設定を確認します。これはシステム全体の設定として設定することも、ブラウザ内で設定することもできます。
- コンピュータにインストールされているセキュリティ ソフトウェアが干渉しているないことを確認します。通常考えられるものとしては、パーソナル ファイアウォール、ウイルス対策ソフトウェアなど、デバイスのセキュリティを「向上させる」ソフトウェア全般が挙げられます。

注:シナリオ 1 またはシナリオ 3 を使用しており(たとえば、dCloud で開発ツールとコードを実行している場合)、dCloud ワークステーションへのアクセスに VPN のみを使用している場合は、Python のテストは省略できます。

Python のインストールのテスト

実施がまだの場合は、Python 用の `virtualenv` の使用を検討してください。これにより、グローバルの Python 環境に影響を与えることなく、テストを容易に実施することができます。

上のページの一番下に [ダウンロード スクリプト(Download Script)] というボタンがあります。このボタンから、Python の簡単なスクリプトがダウンロードできます。コンピュータとそのコンピュータ上の Python 環境が正常に動作している場合、そのスクリプトを実行すると、ラボの VM に接続が戻され、(非常にシンプルな)REST API コールを使用して情報が取得されます。

1. スクリプトをダウンロードし、適切な場所に保存します
2. (利用中のプロットフォーム/Python のバージョンに応じて)`Python connectiontest.py` または、`py -3 connectiontest.py` を使用してスクリプトを実行します。
3. 出力を確認します。

次と同様の出力が表示されれば成功です。

```
$ python ~/Downloads/connectiontest.py


************************************************************
October 12, the Discovery.
```

```
It was wonderful to find America, but it would have been more wonderful to mi
ss

it.

        -- Mark Twain, "Pudd'nhead Wilson's Calendar"


************************************************************

  _____                                _   _
 / ____|                              | | (_)
| |      ___  _ __  _ __    ___  ___| |_ _  ___  _ __
| |     / _ \| '_ \| '_ \ / _ \/ __| __| |/ _ \| '_ \
| |____| (_) | | | | | | |  __/ (__| |_| | (_) | | | |
 \_____\___/|_| |_|_| |_|\___|\___|\__|_|\___/|_| |_|



  ____  _  __
 / __ \| |/ /
| |  | | ' /
| |  | |  <
| |__| | .\
 \____/|_|\_\



$
```

おめでとうございます。VPN 接続とコンピュータの Python 環境が機能しています。

次のページに進み、ラボの開始前に設定するアカウントについての情報を入手します。

自分のコンピュータを設定する方法
このラボでは導入部分を扱います。

ステップ 6：アカウントへのサインアップ

次のアカウントを取得することを推奨します。

Cisco Spark アカウントの取得

(**必須**)ラーニング トラックでは常に、Cisco Spark と呼ばれるシスコのインスタント メッセージ ツールを使用します。Cisco Spark を使用するには、Spark のアカウントが必要です。まだ、Spark のアカウントを持っていない場合は、Cisco Spark に登録します。これは無料のサービスです。登録で必要となるのは、電子メール アドレスのみです。

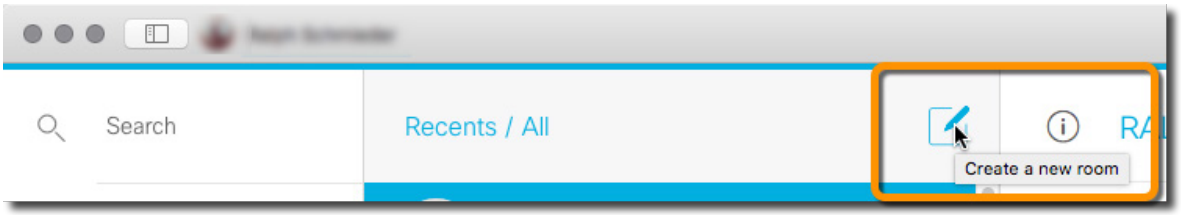
1. https://www.ciscospark.com に移動します。
2. 登録情報を入力します

Spark ルームの作成

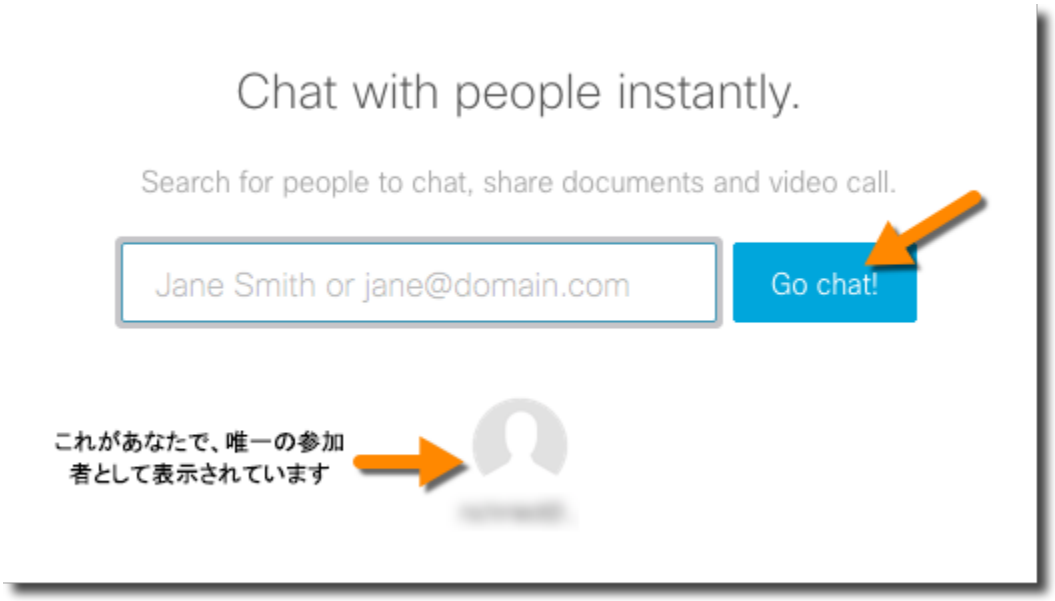
自分自身にメッセージを送付できるよう、Spark にテスト ルームを作成します（特に指定されていない限り。ハンズオン イベントでは、すべての人がメッセージを投稿できる特定のイベント ルーム名が提供されます）。

Spark ルームを作成するには、次の手順に従います。

1. Spark クライアント、または Spark Web クライアントを開きます
2. 新しいルームを作成します。



3. 電子メール アドレスを追加し、「リターン」キーを押します。



4. [チャットを始める（Go chat!）] をクリックし、ルームを作成します。
5. ルームは、メッセージを送信した際にのみ開始されます。
6. メッセージを入力します。ルームに [タイトルなし（Untitled）] と表示されます
7. タイトルをクリックし、「LetMySPARKRoomFly」などの覚えやすい名前に変更します。

おめでとうございます。これで、Spark ルームができました。Spark API に対してテストを行う際、後でこのルームを使用できます。

Cisco CCO アカウントの取得

（必須）Cisco.com の Web サイトで CCO アカウントに登録することを推奨します。CCO アカウントは、特定のラボ環境やリソースにアクセスする際に必要となります。

1. https://tools.cisco.com/RPF/register/register.do に移動します。
2. CCO の登録情報を入力します

DevNet への登録

（オプション）DevNet への登録を強く推奨します。無料のラーニング ラボや DevNet サンドボックスへのアクセスなどの利点が多数あります。

GitHub アカウントの入手

（オプション）必須というわけではありませんが、GitHub への参加を推奨します。

1. https://github.com/join に移動します。
2. 登録情報を入力します。

GitHub アカウントは必須ではありません。このトラックで使用するリポジトリは、Github アカウントを必要としないパブリック リポジトリです。

ただし、Github アカウントがあれば、自身のコード リポジトリを作成することができ、個人のコードも保存できるため、非常に便利です。たとえば、ラボで作成したソース コードを、後で使用できるよう GitHub アカウントに保存することができます。

次の手順

引き続き、事前準備のモジュールの最後のステップである*「イベント前の設定のトラブルシューティング」*に進みます。

自分のコンピュータを設定する方法
このラボでは導入部分を扱います。

ステップ 7：イベント前の設定のトラブルシューティング

これまでの手順を完了させるまでに何か問題はありましたか。

このトラックのラボ実習を進められるようにするには、次の要件を最低限満たすことが必要です。

- Python 環境が正しくインストールされている（開発ツールの実行に、ローカル マシンを使用している場合）
- 次のいずれかでのリモート接続用クライアント
    - VPN がインストール済み（AnyConnect または OpenConnect）、または、
    - HTML5 対応ブラウザ
- CCO アカウント
- ラボ環境への接続がテスト済み

ハンズオン イベントの際には、トラブルシューティングに多くの時間が割けないことに注意してください。このため、イベントの**開始前に**、Python と AnyConnect が正常に動作していることを確認する必要があります。

ハンズオン イベントへの参加を計画していて、上記の内容が機能していない場合は、ハンズオン イベントの際に Web ベースのリモート デスクトップ セッションを使用することを**強く**推奨します。Web ベースのリモート デスクトップにより、ローカル マシンに Python をインストールすることなく、完全なラボ環境にアクセスすることができます。

サポートの利用

Python 環境が機能**しない**場合は、http://stackoverflow.com で一般的な質問への回答を検索できます。

DevNet サポートの Spark ルームで、設定について質問することができます。DevNet チームがそのルームの中で、直接質問に回答します。

これでイベント前の設定に関する説明を終了します。DevNet リソースの次のモジュールを開始する準備が整いました。