



Kompira Cloud Sonarチュートリアル



株式会社フィックスポイント

チュートリアル内容

- Sonarを登録するところから始めて、構成情報を収集するまでを見ていくチュートリアルとなります

内容

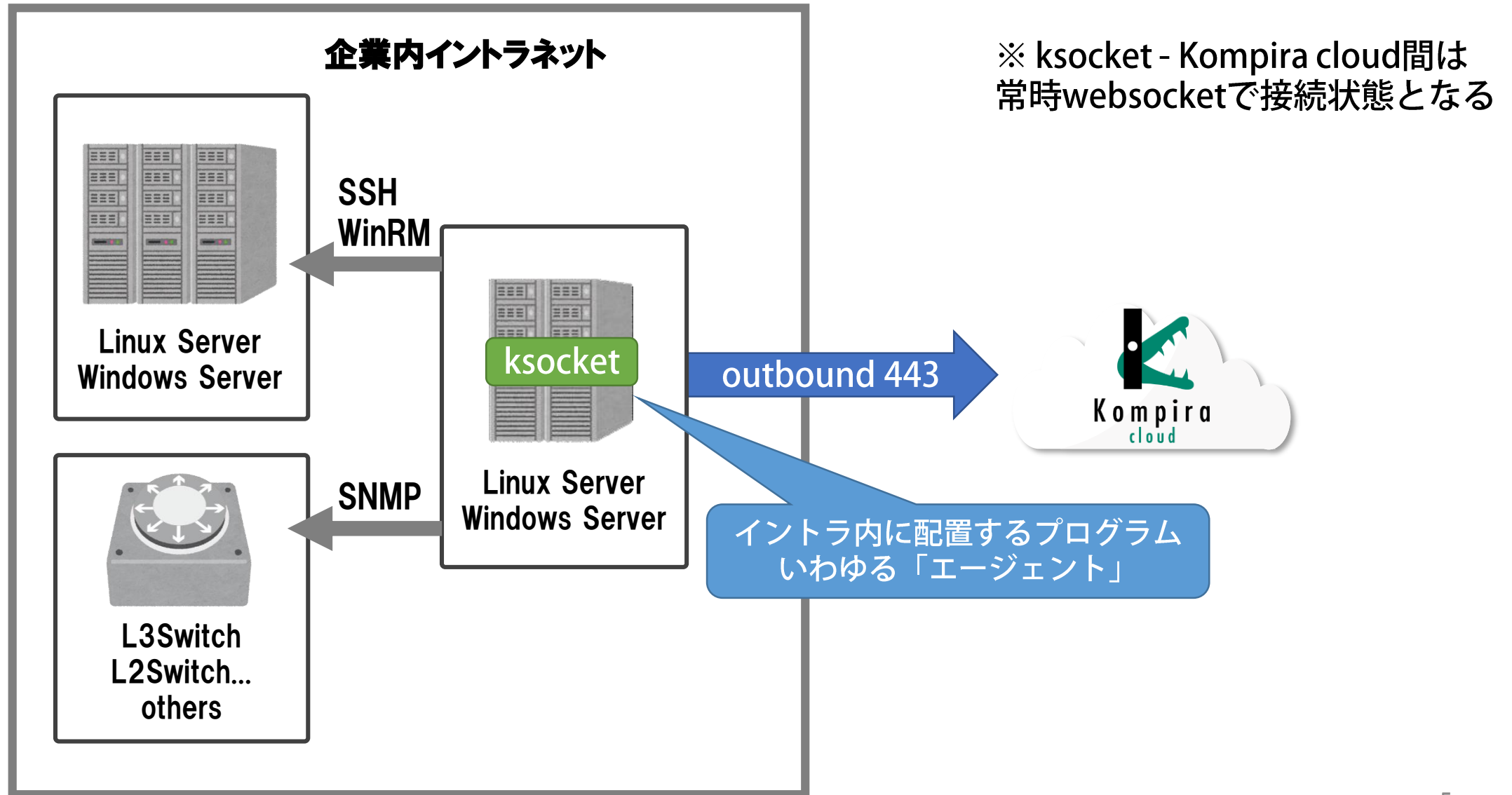
1. Sonarの構成
2. スペース登録をする
3. ksocketのセットアップ
4. スキャンを実行してみる
5. アカウント情報を設定してスキャンできる情報を増やす
6. スキャンした結果をAPIで取得する

1. Sonarの構成

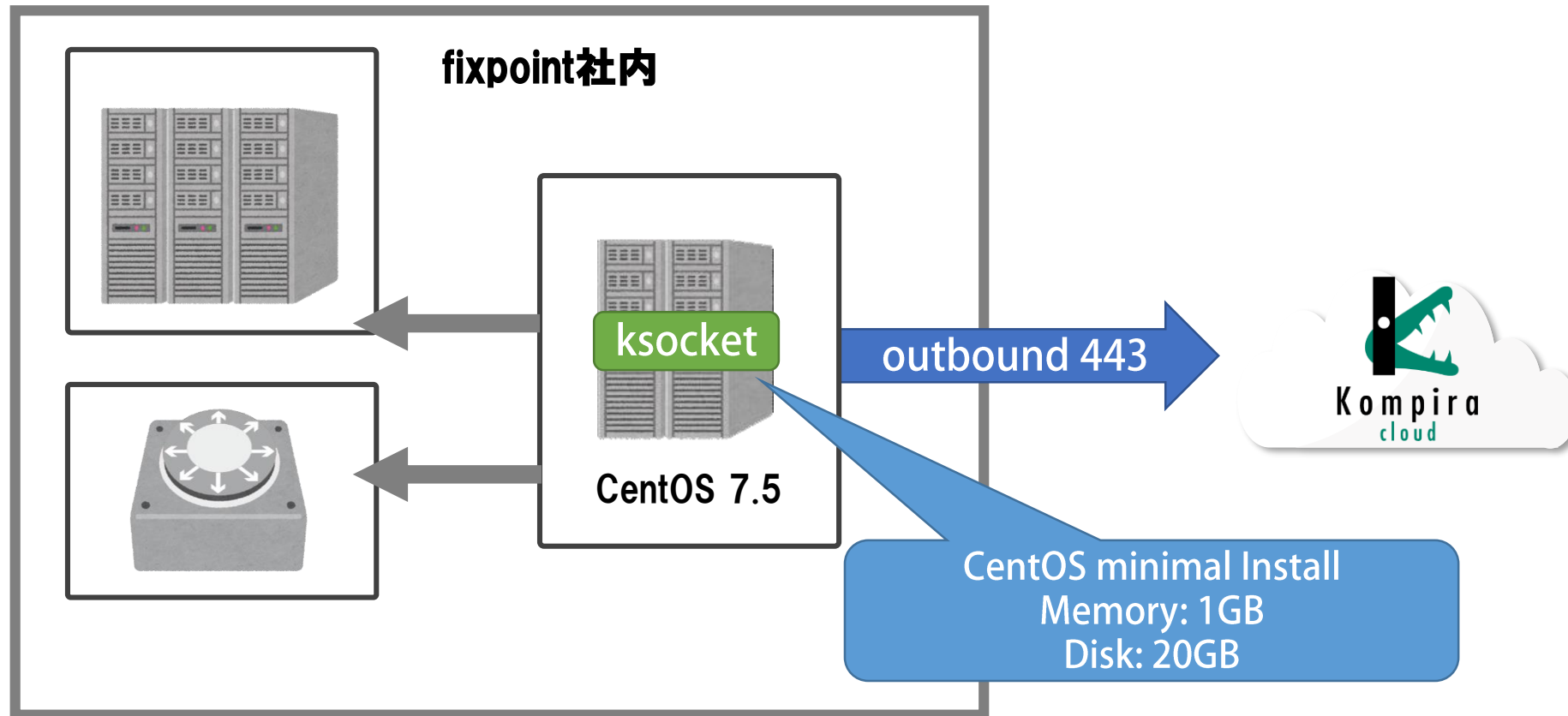
Sonarはクラウドサービスです。

通常のコラウドサービスに加えて、
ksocketというモジュールを使うという独特の点があります。
はじめての章では、Sonarの全体像を簡単に確認しておきましょう。

Sonarの構成



本チュートリアルで前提とする構成



- 本チュートリアルではksocketのインストール先としてCentOS 7.5を使用します

2. スペース登録をする

まずはSonar上に自分が管理する場所、「スペース」を作成する必要があります。
ここではスペースの作成方法と、この後のksocketセットアップのための準備を行います。

スペース登録をする

- 以下URLより新しいスペースを作成
 - <https://register.cloud.kompira.jp>

A screenshot of the Kompira cloud registration page. The page has a light purple header with the Kompira cloud logo (a green 'K' with a dinosaur head) and the text 'Kompira cloud'. Below the header, it says 'Kompira cloudをはじめましょう！' and '無料でお試しください。まずはあなたのスペースを作成しましょう。'. The main content area is white and contains four input fields: 'スペースID' (with a help icon and example 'fixpoint'), 'スペース表示名' (with example '株式会社フィックスポイント'), 'ユーザー表示名' (with example '不動 点三郎'), and 'メールアドレス'.

Kompira cloud

Kompira cloudをはじめましょう！

無料でお試しください。まずはあなたのスペースを作成しましょう。

スペースID ⓘ

例: fixpoint

スペース表示名

例: 株式会社フィックスポイント

ユーザー表示名

例: 不動 点三郎

メールアドレス

Kompira cloudにログイン

- <https://xxxxxxxx.cloud.kompira.jp>



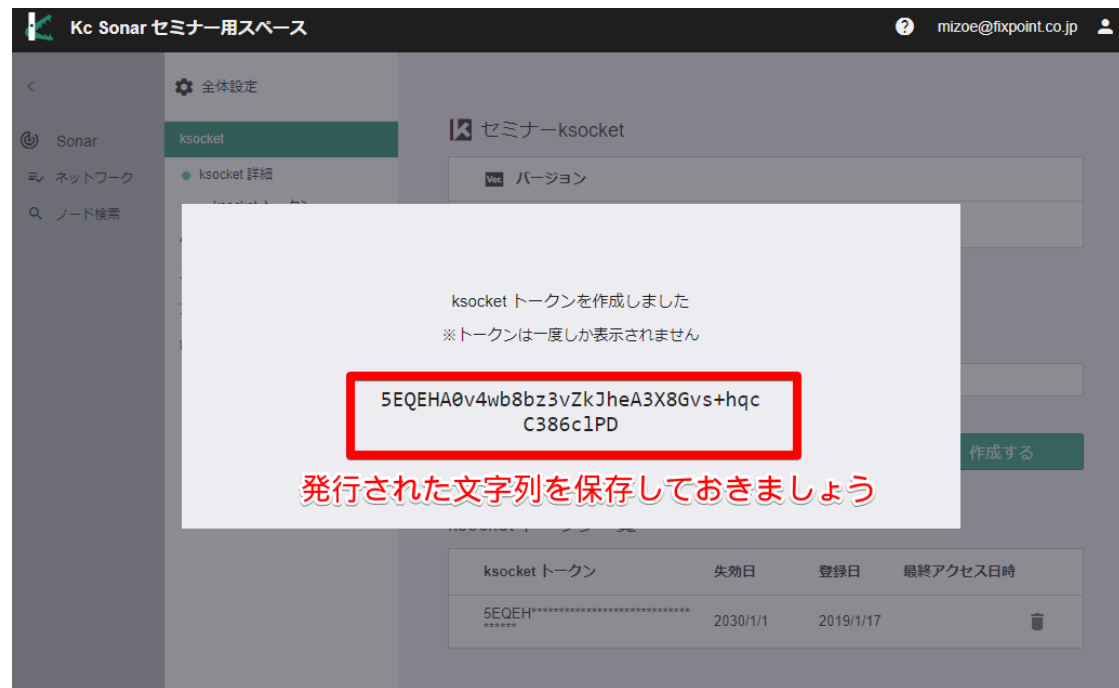
ksocketスロットの作成とトークン発行 (1)

- ksocketが接続する先を「ksocketスロット」と言います
- Kompira cloudの全体設定より「ksocket」を選択し、新しくスロットを作成しましょう



ksocketスロットの作成とトークン発行 (2)

- ksocketスロットを作成したら、そこから「ksocketトークン作成」をしましょう
- 表示された文字列を保存しておきます



3. ksocketのセッ トア ッ プ

Sonarは構成情報を管理するためのサービスであり、ksocketというモジュールを自分の環境にインストールすることで構成情報を収集することができます。

ここではksocketのインストールと初期設定の方法を見ていきましょう。

ksocketのセッ トア ッ プ (1)

- ksocketのダウンロード
 - 以下より最新のksocketを取得
<https://blog.cloud.kompira.jp/entry/downloads>
- ksocketインストール
 - サーバ上にksocketファイルを配置し、以下のコマンドを実行

```
$ tar zxf ksocket-linux-64-20190510.0.tar.gz  
$ sudo ./ksinstall
```

```
$ sudo ./ksinstall
```

(略)

Please confirm or refill the ksocket installation directory (Prefix):

Prefix (ksocket installation directory): `/opt/fixpoint/ksocket`

インストール先ディレクトリを入力

Please confirm informations below for connecting to your space in the Kompira cloud:

Host (e.g. fixpoint.cloud.kompira.jp): `xxxxxxxx.cloud.kompira.jp`

登録したスペースのURLを入力

Token (<https://seminarhoge.cloud.kompira.jp/preference/ksocket/ksockets>): `F39or7A71NQmnv2GaeZ3yidEo/80YgGT38f+thQ2`

Installing ksocket into /opt/fixpoint/ksocket ...

Kompira cloud上で発行した
ksocketトークン文字列を入力

ksocketのセットアップ (2)

(略)

```
Installing ksocket service/daemon ...
```

```
    Registering ksocket service/daemon ...
```

```
Creating uninstaller ...
```

```
    Creating uninstaller as /opt/fixpoint/ksocket/uninstaller.sh ...
```

```
    Making the uninstaller executable ...
```

```
Thanks for installing ksocket, Fixpoint. Inc.
```

- 上記のような表示がされていればksocketのインストール完了

ksocketのセッ トアップ (3)

- Kompira cloudのksocketスロット画面で「接続済」になっていることを確認しましょう



The screenshot shows the 'Kc Sonar セミナー用スペース' (Kc Sonar Seminar Space) interface. The left sidebar contains navigation links: '全体設定' (Overall Settings), 'ksocket', 'ksocket 詳細' (ksocket Details), 'ksocket トークン' (ksocket Token), 'API トークン' (API Token), 'スペース' (Space), 'プロフィール' (Profile), and '解約' (Cancellation). The main content area is titled 'セミナーksocket' (Seminar ksocket) and includes a 'バージョン' (Version) of 1.4.3. The '接続状況' (Connection Status) is shown as '接続済' (Connected), which is highlighted with a red box. Below this, a table lists network interfaces with their MAC addresses, IP addresses, and netmasks. At the bottom, there is a section for '関連情報' (Related Information) with a link to 'ksocket トークン' (ksocket Token).

インターフェース名	MAC アドレス	IP アドレス	ネットマスク
lo	00:00:00:00:00:00	127.0.0.1	255.0.0.0
lo	00:00:00:00:00:00	::1	ffff:ffff:ffff:ffff:ffff:ffff/128
enp0s3	08:00:27:d1:a0:76	10.10.0.107	255.255.0.0
enp0s3	08:00:27:d1:a0:76	fe80::a570:c9ee:6142:eb7e%enp0s3	ffff:ffff:ffff:ffff::/64

ksocketの設定

- ksocketの設定を確認・変更する

```
$ cd /opt/fixpoint/ksocket  
  
$ sudo ./bin/ksocket config <--- 設定項目をすべて表示する  
  
$ sudo ./bin/ksocket config loglevel <--- loglevelの項目を表示する  
  
$ sudo ./bin/ksocket config loglevel DEBUG <--- loglevelの項目をDEBUGに変更
```

詳細なログが出力されるようにする
デフォルトでは設定されていませんが、今回のチュートリアルでは追加します

ksocketの操作

```
$ sudo systemctl status ksocket    <---- ksocketの状態を確認する  
$ sudo systemctl start ksocket     <---- ksocketを起動する  
$ sudo systemctl stop ksocket      <---- ksocketを停止する  
$ sudo systemctl restart ksocket   <---- ksocketを再起動する
```

- ksocket configで値を更新した場合、反映には再起動が必要です

ksocketのログを確認

- ksocketのログファイル

/opt/fixpoint/ksocket/var/log/ksocket/ksocket.log

```
$ cd /opt/fixpoint/ksocket
$ tail var/log/ksocket/ksocket.log
2019-01-11T11:31:47+0900  DEBUG  ksocket.connection.websocket:_heartbeat:151  Heartbeat ping send with timeout 30 seconds
2019-01-11T11:31:47+0900  DEBUG  ksocket.connection.websocket:_heartbeat:158  Heartbeat pong has received. Wait 60 seconds
2019-01-11T11:32:48+0900  DEBUG  ksocket.connection.websocket:_heartbeat:151  Heartbeat ping send with timeout 30 seconds
2019-01-11T11:32:48+0900  DEBUG  ksocket.connection.websocket:_heartbeat:158  Heartbeat pong has received. Wait 60 seconds
2019-01-11T11:33:48+0900  DEBUG  ksocket.connection.websocket:_heartbeat:151  Heartbeat ping send with timeout 30 seconds
2019-01-11T11:33:48+0900  DEBUG  ksocket.connection.websocket:_heartbeat:158  Heartbeat pong has received. Wait 60 seconds
2019-01-11T11:34:50+0900  DEBUG  ksocket.connection.websocket:_heartbeat:151  Heartbeat ping send with timeout 30 seconds
2019-01-11T11:34:50+0900  DEBUG  ksocket.connection.websocket:_heartbeat:158  Heartbeat pong has received. Wait 60 seconds
2019-01-11T11:35:51+0900  DEBUG  ksocket.connection.websocket:_heartbeat:151  Heartbeat ping send with timeout 30 seconds
2019-01-11T11:35:51+0900  DEBUG  ksocket.connection.websocket:_heartbeat:158  Heartbeat pong has received. Wait 60 seconds
```

Kompira cloudとksocketが疎通状態かどうかを
定期的に確認していることを示すログ
これが出力されていれば正常に接続できています

プロキシサーバ経由でksocketを動かす場合

- ksocketとKompira cloudを接続する際にプロキシサーバを経由する必要がある場合、プロキシ用モジュール ksbridge を使用してください
- 詳細は以下よりご確認ください
 - <https://blog.cloud.kompira.jp/entry/manual/ksocket-with-proxy>

AWS / Azureでksocketを動かす場合

- AWS

- ksocketがインストールされたインスタンスに対して、AWS REST APIのReadOnlyAccess権限を設定する必要があります。

- Azure

- Service Principalを作成し、Azure REST APIのReader権限を設定する必要があります。
- Service PrincipalのクライアントID, 認証キー, テナントIDを設定したtomlファイルを作成する必要があります。

`/opt/fixpoint/ksocket/etc/ksocket/azure.toml`

- より詳しい設定方法はksocketドキュメントを参照してください
<https://blog.cloud.kompira.jp/entry/downloads>

4. スキャンを実行してみる

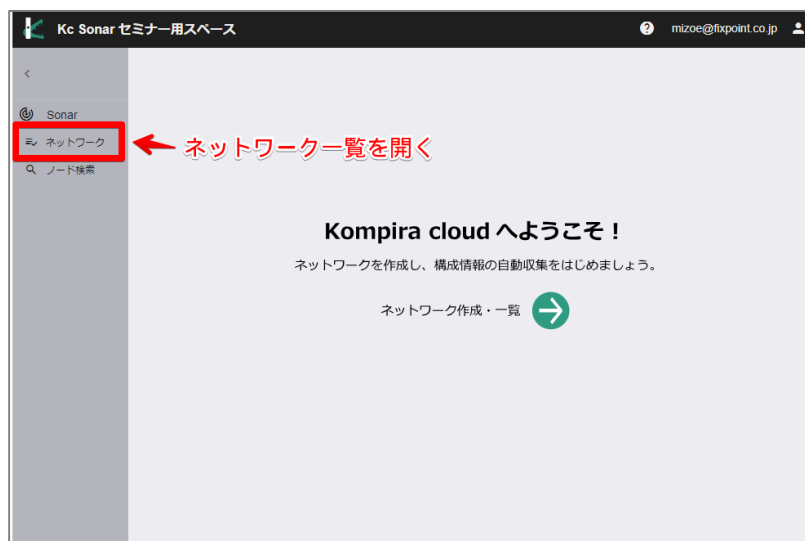
ここまでで、Sonarのスペースとksocketの準備ができました。

それでは、さっそく構成情報の収集を行ってみましょう。

Sonarでは、構成情報を収集することを「スキャン」と呼びます。
ここでは、スキャンの実行方法と、結果の見方について解説します。

「ネットワーク」を作成する

- スキャンで収集した構成情報を保存する先を「ネットワーク」と呼びます
- 「ネットワーク一覧」より、新しいネットワークを作成しましょう

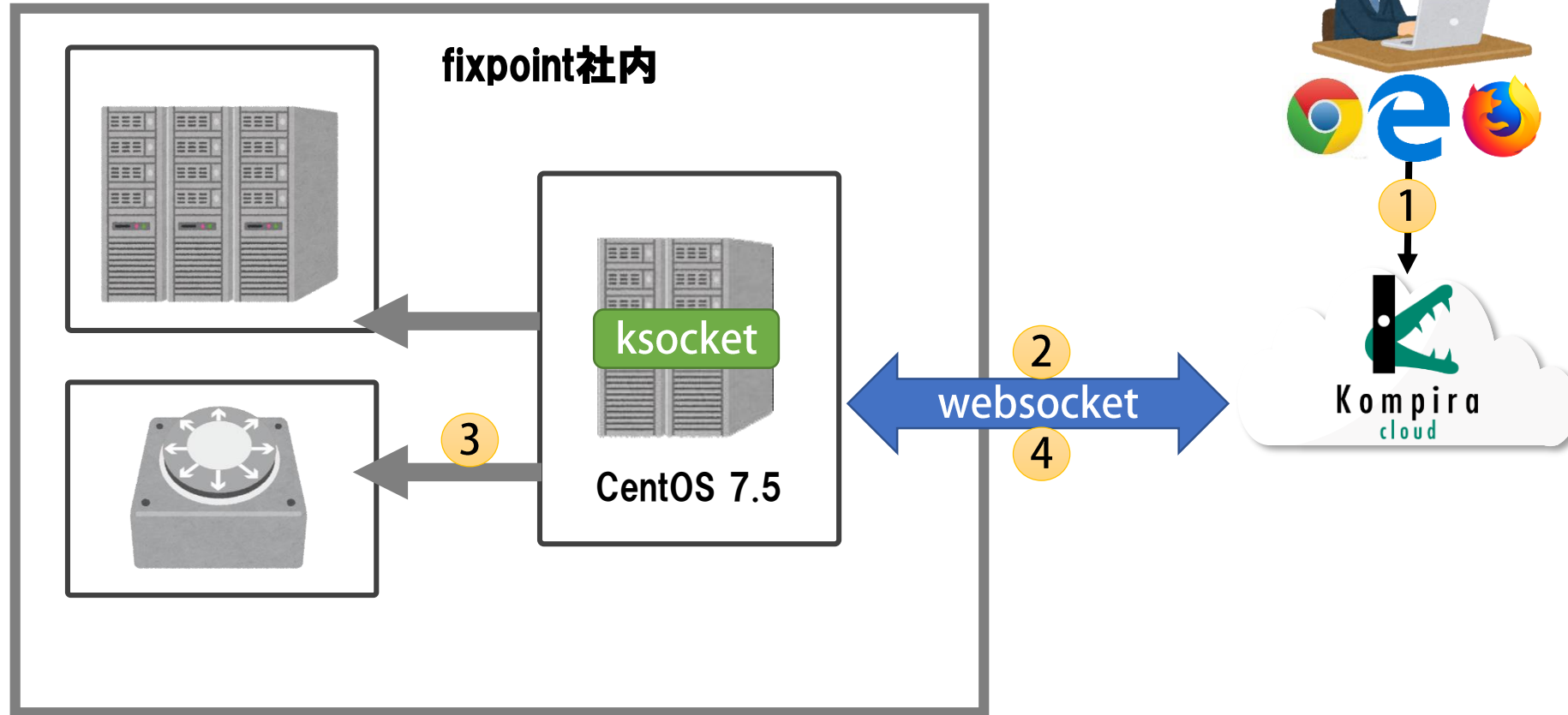


スキャンを実行する

- ksocketが構成情報を収集することを「スキャン」と呼びます
- 作成したネットワークを選択し、「スキャン」画面で実行することができます



スキャンの流れ



1. Sonar上でスキャンの実行をする
2. Sonarからksocketにスキャン実行指示を送る
3. ksocketが自分のいるネットワークを探索して構成情報を収集する
探索の詳しい解説: <https://blog.cloud.kompira.jp/entry/manual/scan-example>
4. ksocketが収集したデータをSonarに送る

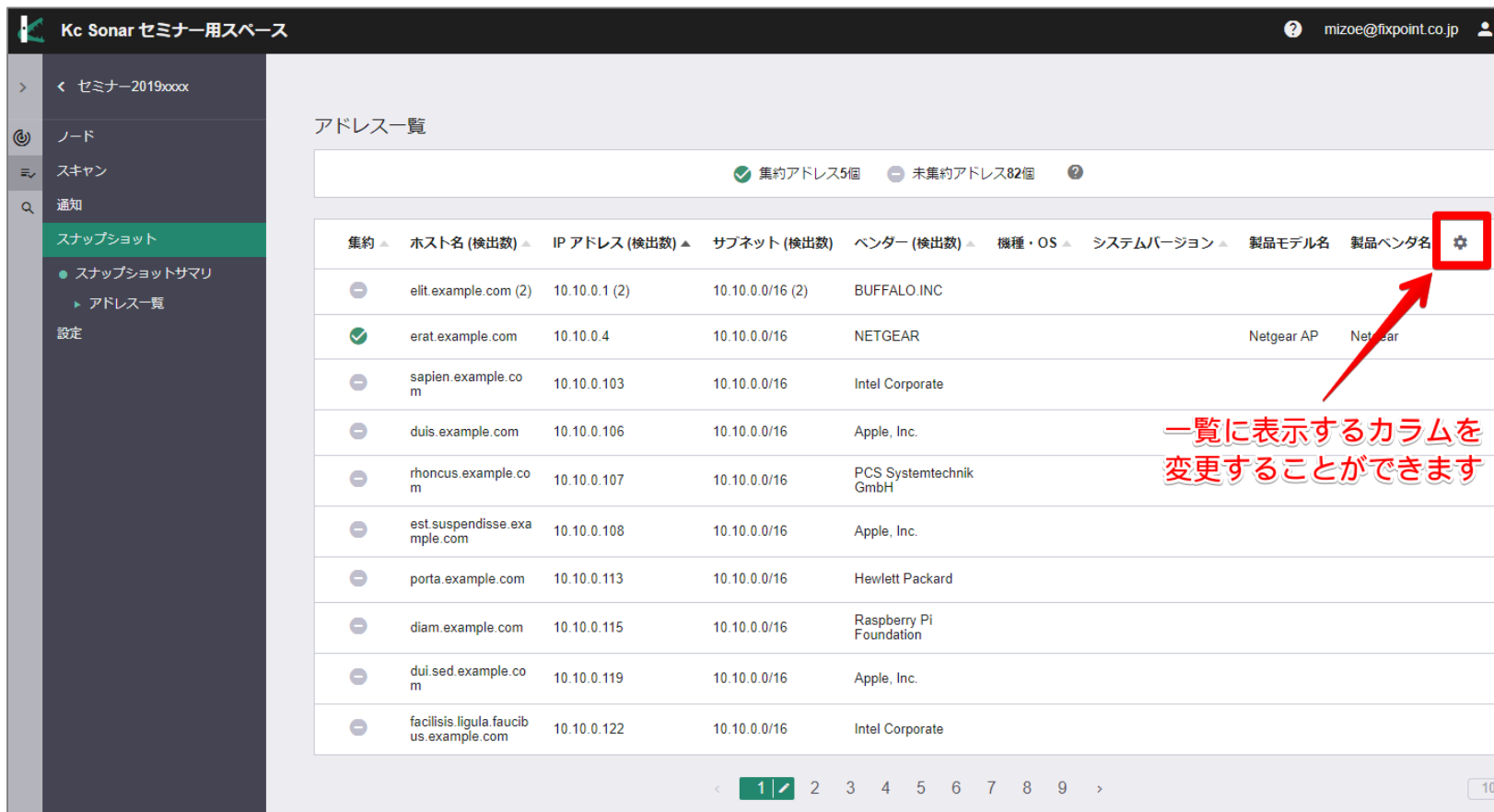
スナップショット

- 1回のスキャンの結果を「スナップショット」と呼びます
- Sonar上で、過去のスナップショットの情報を確認することができます



スナップショットの結果を見る

- 1回のスキャンで取得した構成情報を見ることができます



Kc Sonar セミナー用スペース

mizoe@fixpoint.co.jp

セミナー2019xxxx

ノード

スキャン

通知

スナップショット

スナップショットサマリ

アドレス一覧

設定

アドレス一覧

集約アドレス5個 未集約アドレス82個

集約	ホスト名 (検出数)	IP アドレス (検出数)	サブネット (検出数)	ベンダー (検出数)	機種・OS	システムバージョン	製品モデル名	製品ベンダ名	
—	elit.example.com (2)	10.10.0.1 (2)	10.10.0.0/16 (2)	BUFFALO.INC					
✓	erat.example.com	10.10.0.4	10.10.0.0/16	NETGEAR			Netgear AP	Netgear	
—	sapien.example.com	10.10.0.103	10.10.0.0/16	Intel Corporate					
—	duis.example.com	10.10.0.106	10.10.0.0/16	Apple, Inc.					
—	rhoncus.example.com	10.10.0.107	10.10.0.0/16	PCS Systemtechnik GmbH					
—	est.suspendisse.example.com	10.10.0.108	10.10.0.0/16	Apple, Inc.					
—	porta.example.com	10.10.0.113	10.10.0.0/16	Hewlett Packard					
—	diam.example.com	10.10.0.115	10.10.0.0/16	Raspberry Pi Foundation					
—	dui.sed.example.com	10.10.0.119	10.10.0.0/16	Apple, Inc.					
—	facilisis.ligula.faucibus.example.com	10.10.0.122	10.10.0.0/16	Intel Corporate					

一覧に表示するカラムを変更することができます

1 2 3 4 5 6 7 8 9 10

集約アドレス

- スキャンで取得した各機器情報のうち、以下の条件を満たしたものは「集約アドレス」となります
 - ネットワーク機器
 - SNMPでシリアル番号が取得できた
 - Windows
 - WinRMログインしてシリアル番号が取得できた
 - Linux
 - sshログインして以下のファイルが存在することが確認できた or 作成できた
/var/tmp/cloud.kompira.jp/footprint_xxxx
(存在しない場合はksocketがsshログイン時に作成します)

ノード

- 前ページの「集約アドレス」は、自動的に「ノード」になります
- ノードの情報はスキャンするごとに自動で更新されます

5. スキャンできる情報を増やす

最初のスキャンが無事に完了しました。

ksocketにはまだ何の設定も行っていないので、この時点では最低限の情報しか取得することができません。

そこで、この章ではksocketにアカウント情報を設定して、より多くの情報を取得できるようにする方法を学びましょう。

アカウント情報の設定 (1)

- ksocketは以下の形式によるアクセスに対応しています
 - SNMP (v2c / v3)
 - SSH
 - WinRM
- アカウント情報設定ディレクトリ
`/opt/fixpoint/ksocket/etc/ksocket/credentials`

アカウント情報の設定 (2)

- アカウント設定ディレクトリの構成 (初期状態)

```
/opt/fixpoint/ksocket/etc/ksocket/credentials/
```

```
├── snmp
│   ├── 999-default.toml
│   └── 999-example.toml.skeleton
├── ssh
│   └── 999-example.toml.skeleton
└── winrm
    └── 999-example.toml.skeleton
```

snmp (v2c) publicコミュニティの設定を
デフォルトで配置している

- 1つのアカウント情報ごとに1つのtomlファイルを作成します
- .skeleton は設定の書き方の例として配置しているダミーファイル
- 新しいアカウント情報を設定するときは、ダミーファイルをコピーして作成しましょう

アカウント情報の設定 (SNMP v2c)

```
$ cd /opt/fixpoint/ksocket/etc/ksocket/credentials/snmp
$ cp 999-example.toml.skeleton 100-test.toml
$ vi 100-test.toml
```

```
includes = ["10.10.0.0/24",
            "10.20.0.0/24"]
```

includes: アカウント情報を使う対象のIPアドレス
ksocketがスキャンでIPアドレスを見つけたとき、ここに含まれて
いればアカウント情報を使用してアクセスする(※)

```
port = 161
```

```
[authData]
community: "public"
```

snmpのコミュニティ名

(※) includesに設定したIPアドレスすべてにksocketからスキャンしに行くわけではないことに注意してください。
例えば「10.0.0.0/8」と書いても、このネットワークすべてにksocketがアクセスするわけではありません。

includesに「0.0.0.0/0」と設定すると、「見つかったすべてのIPアドレスでこのアカウント情報を使用する」の意味になります。

アカウント情報の設定 (SNMP v3)

```
$ cd /opt/fixpoint/ksocket/etc/ksocket/credentials/snmp
$ cp 999-example.toml.skeleton 200-v3test.toml
$ vi 200-v3test.toml
```

```
includes = ["10.10.0.0/24",
            "10.20.0.0/24"]
```

```
port = 161
```

```
[authData]
username = "snmp-user"
```

```
authProtocol = "usmHMACMD5AuthProtocol"
authKey = "your-password"
```

```
privProtocol = "usmAesCfb128Protocol"
privKey = "priv-password"
```

認証方式とパスワードの指定

authProtocolには以下のいずれかを指定

認証をしない場合: "usmNoAuthProtocol"

MD5を使用する場合: "usmHMACMD5AuthProtocol"

SHAを使用する場合: "usmHMACSHAAuthProtocol"

暗号化方式とパスワードの指定

privProtocolには以下のいずれかを指定

暗号化をしない場合: "usmNoAuthProtocol"

DESを使用する場合: "usmDESPrivProtocol"

AESを使用する場合: "usmAesCfb128Protocol"

アカウント情報の設定 (SSH) (1)

- パスワードログインの場合

```
$ cd /opt/fixpoint/ksocket/etc/ksocket/credentials/ssh  
$ cp 999-example.toml.skeleton 100-test.toml  
$ vi 100-test.toml
```

```
includes = ["10.10.0.0/24"]
```

```
port = 22
```

```
[account]  
username = "john"  
password = "password"
```

ssh接続するユーザ名、パスワード

アカウント情報の設定 (SSH) (2)

- 鍵ファイルログインの場合

```
$ cd /opt/fixpoint/ksocket/etc/ksocket/credentials/ssh
$ cp 999-example.toml.skeleton 200-keytest.toml
$ vi 200-keytest.toml
```

```
includes = ["10.10.0.0/24"]
```

```
port = 22
```

```
[account]
username = "john"
```

```
[[account.clientKeys]]
filename = "../../id_rsa.common"
passphrase = "pasufureezu"
```

鍵ファイルのパスとパスフレーズを指定
鍵ファイルはksocketサーバに配置する

アカウント情報の設定 (WinRM) (1)

- WinRM: Windows Remote Management
 - Windowsに対してリモートアクセスする際に使用する仕組み
- WinRMはデフォルトでは有効でないため、Windows(アクセスされる側)で設定する必要があります

<https://blog.cloud.kompira.jp/entry/manual/winrm-settings>

アカウント情報の設定 (WinRM) (2)

```
$ cd /opt/fixpoint/ksocket/etc/ksocket/credentials/winrm  
$ cp 999-example.toml.skeleton 100-test.toml  
$ vi 100-test.toml
```

```
includes = ["10.10.0.0/24"]
```

```
port = 5985
```

WinRMをデフォルトで設定すると5985ポートを使用する

```
authMethod = "ntlm"
```

認証形式

```
[account]
```

```
username = "john"
```

```
password = "password"
```

"basic" "ntlm" "credssp" のいずれかを選択できる
特にこだわりがなければntlmを設定しましょう

複数のアカウント情報を設定する

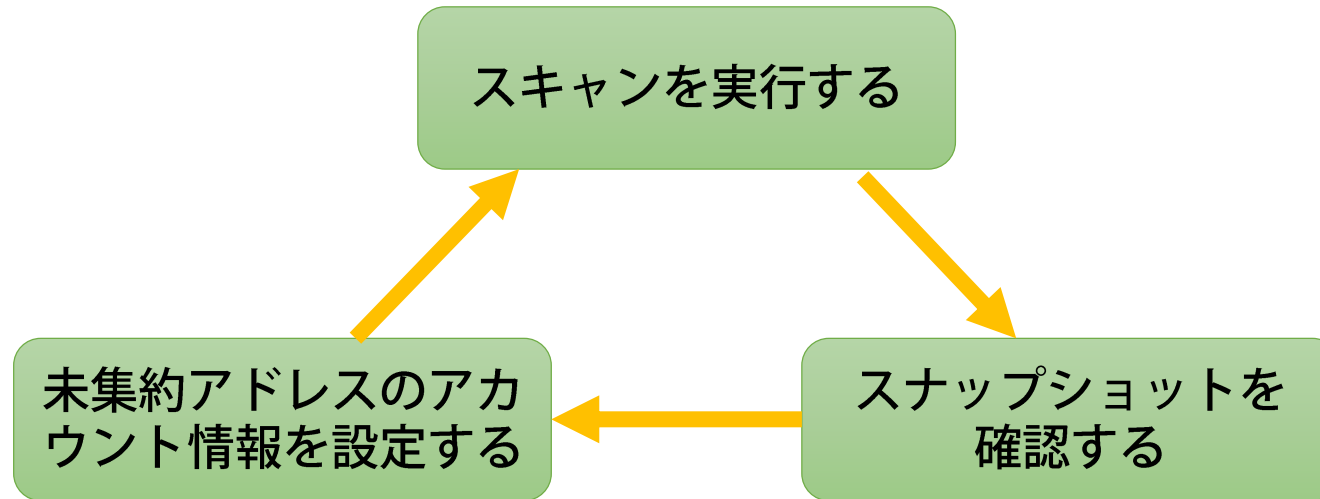
```
ssh
├── 100-test1.toml
├── 200-test2.toml
├── 300-test3.toml
└── 999-example.toml
```

- 複数のtomlファイルが存在する場合、ファイル名の辞書順にアクセス試行し、アクセスに成功したところで終了します
- 上記の場合、以下のような順序で処理をします
 - 100-test1.toml のアカウントを使ってsshアクセス試行する
 - 200-test2.toml のアカウントを使ってsshアクセス試行する
 - 300-test3.toml のアカウントを使ってsshアクセス試行する
 - 999-example.toml のアカウントを使ってsshアクセス試行する

スキャンを実行する

- アカウント情報の設定を行った後、再度 Sonar からスキャンを実行してみましょう
- 設定したアカウントを持つサーバやネットワーク機器の情報が増えているか確認してみましょう

環境設定の流れ



- このサイクルを繰り返すことで、ノードを増やしていきましょう

6. スキャンした結果をAPIで取得する

これで構成情報を収集でき、その結果をブラウザ上で見るできるようになりました。

Sonarでは、ブラウザ上でできることは全てAPIで行うことができます。

curlコマンドを使って、APIで情報を取得してみましょう。

APIトークンの発行

- 「全体設定 > APIトークン」より、「ksocketトークン作成」をしましょう
- 表示された文字列を保存しておきます



curlコマンドによるAPI実行 (1)

- ネットワーク一覧をAPIで取得する例

```
$ sudo yum install epel-release  
$ sudo yum install jq  
$ curl -s -X GET -H "accept: application/json" -H "X-Authorization: Token IqyzDqV+9Her3+FfgXAQye  
4P4dksgpYgZz/OQzP1" "https://xxxxxxxx.cloud.kompira.jp/api/apps/sonar/networks" | jq .
```

Kompira cloud上で発行したAPI
トークン文字列を指定

```
$ sudo yum install epel-release
$ sudo yum install jq
$ curl -s -X GET -H "accept: application/json" -H "X-Authorization: Token IqyzDqV+9Her3+FfgXAQye
4P4dksgpYgZz/OQzP1" "https://xxxxxxx.cloud.kompira.jp/api/apps/sonar/networks" | jq .
{
  "offset": 0,
  "limit": 30,
  "total": 1,
  "items": [
    {
      "networkId": "d75664e2-c921-4c91-8abc-3b52c9f84449",
      "displayName": "Kcチュートリアルxxxx",
      "configuration": {
        "0c84549d-556a-4d14-8376-f831e599864b": {}
      },
      "createdAt": "2019-01-23T04:06:00Z",
      "createdBy": "1c31d0ef-78d1-471f-a3b6-97aa0bf8b4fd",
      "updatedAt": "2019-01-23T04:10:13Z",
      "updatedBy": "1c31d0ef-78d1-471f-a3b6-97aa0bf8b4fd",
      "lastScanSucceededAt": "2019-01-23T04:17:02Z",
      "numberOfNewNodes": 5,
      "numberOfStableNodes": 0,
      "numberOfOutdatedNodes": 0
    }
  ]
}
```

curlコマンドによるAPI実行 (2)

- あるネットワークのノード一覧をAPIで取得する例

```
$ curl -s -X GET -H "accept: application/json" -H "X-Authorization: Token IqyzDqV+9Her3+FfgXAQye  
4P4dksgpYgZz/OQzP1" "https://xxxxxxx.cloud.kompira.jp/api/apps/sonar/networks/de538204-1184-450  
2-a5ea-c56625a7b506/managed-nodes"
```

- あるノード情報をAPIで取得する例

```
$ curl -s -X GET -H "accept: application/json" -H "X-Authorization: Token IqyzDqV+9Her3+FfgXAQye  
4P4dksgpYgZz/OQzP1" "https://xxxxxxx.cloud.kompira.jp/api/apps/sonar/networks/de538204-1184-450  
2-a5ea-c56625a7b506/managed-nodes/a7c32742-d172-4b56-a671-e3582e8559d1"
```

API ドキュメント

- 全てのAPIはドキュメントで確認することができます
 - <https://cloud.kompira.jp/docs/apidoc>



おわりに

本チュートリアルは以上で終了となります。
何かわからないことがある場合は、以下からお問合せください。

<https://www.fixpoint.co.jp/cloudcontact/>
メール: support@kompira.jp