OPNFVプロジェクトについて

沖縄オープンラボラトリ 研究グループ
仲間　修也
2016/10/14

自己紹介

- 名前
  - 仲間　修也（なかま　しゅうや）
- 所属
  - NECソリューションイノベータ株式会社
  - 2014年1月より沖縄オープンラボラトリにて常駐勤務
- 業務経歴
  - 入社後はカメラシステムや携帯開発（Android）をやってきました
  - 沖縄オープンラボラトリでは
    - 昨年度までOpenFlowパッチパネル（OF-Patch）の研究・開発を行ってきました。
      - OpenFlowでパッチパネルを実現したSDNアプリケーション
    - 本年度5月よりOPNFVを触りはじめています。

目次

- 沖縄オープンラボラトリ(以下、OOL)紹介・活動方針
- OPNFVとは
- OOLのOPNFVの取り組み
  - Pharos Test Lab構築
  - OOLとしてのOPNFV評価活動
    （テストプロジェクトを動作させて結果のまとめ）
    - YardStick
    - Functest
  - VNFテスト基盤のPoC開発

組織概要

- 名称
  - 一般社団法人　沖縄オープンラボラトリ　（略称：沖縄オープンラボ、OOL、等）
- 設立
  - 2013年5月8日　NTTコミュニケーションズ、NEC 、イイガの3社にて設立
- 所在地
  - 〒904-2241　沖縄県うるま市字兼箇段61番地1
    沖縄情報通信センター　ビジネス棟201号
  - 電話：098-989-1940

  - Web：　http://www.okinawaopenlabs.org/
  - Facebook：　http://www.facebook.com/okinawaopenlabs/
- 代表
  - 理事長　伊藤 幸夫
- 目的
  - 情報通信における先進技術（次世代ICT基盤技術）の実用化、普及のための研究開発活動　⇒クラウドとSDNの検証活動を実施

活動方針

- 次世代ICT基盤技術として、情報通信において大きな潮流となっている**クラウド技術とSDN/NFV技術**に着目、より柔軟で使い易いICT基盤の実現を目指し、これらの融合に取り組む
  - 対象技術領域（クラウド＋SDN/NFV）
    - クラウドコンピューティング（OpenStack）
    - 次世代型ネットワーク（SDN/NFV）
- 活動拠点を沖縄に置くことによりおきなわSmart Hub構想実現に貢献する
  - 国内外の企業・ビジネス・人材が活発に交流・集積する拠点を沖縄に形成
  - **人材の育成**に取り組み、人材の高度化に貢献
  - 国際的な情報発信、意見交換、人的交流の場として、**国際会議**等を主催し、研究拠点、ビジネス拠点としての**沖縄の国際的知名度向上、国際的誘引力強化**に貢献する

沖縄オープンラボラトリ会員一覧
（総会員数：43　正会員：5　賛助会員：21　特別会員：17）

2016年9月現在

総会員数: 43
正 会 員: 5
賛助会員: 21
特別会員: 17

正会員

- NTTコミュニケーションズ
- 日本電気
- イイガ
- 伊藤忠テクノソリューションズ
- 富士通

賛助会員

- NTTPCコミュニケーションズ
- イーサイド
- NECソリューションイノベータ
- イクシアジャパン
- ブロケードコミュニケーションズシステムズ
- QualiSystems
- アラクサラネットワークス
- ブロードバンドタワー
- 日本ラドウェア
- 新日鉄住金ソリューションズ
- 日本シエナコミュニケーションズ
- オーシーシー
- アドックインターナショナル
- 沖縄データセンター
- オキット
- シスコシステムズ
- モノビット
- 日商エレクトロニクス
- Arbor Networks
- inwinSTACK
- TIS

特別会員

- 琉球大学工学部
- 国際電子ビジネス専門学校
- 沖縄県立未来工科高等学校
- Institute for Information Industry
- 慶應大学理工学部
- 東京大学大学院情報学環
- 神戸情報大学大学院
- ETRI
- 沖縄ゲーム企業コンソーシアム
- Open Networking Foundation
- 国立研究開発法人情報通信研究機構
- Malaysian Institute of Information Technology, Universiti Kuala Lumpur
- OpenDaylight
- 特定非営利活動法人エルピーアイジャパン
- 九州産業大学　情報科学部
- D4Cloud コンソーシアム
- 沖縄工業高等専門学校

□ 沖縄県内から参加の会員（8企業・団体）　□ アジアから参加の会員（3か国・地域、4企業・団体）

OPNFVとは

OPNFVとは

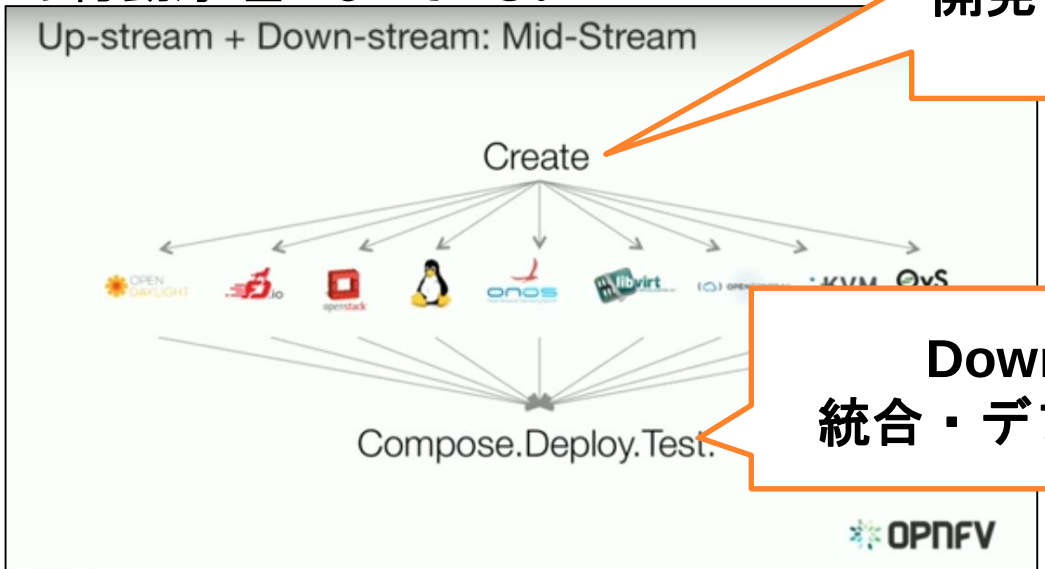
- OPNFV(Open Platform for NFV、www.opnfv.org)とはオープンソースによるNFVリファレンスプラットフォームの実現を目指すプロジェクトである
- 2014年10月に設立され、参加企業は現在57社（10月現在）にのぼる。※https://www.opnfv.org/about/memberslist
- OPNFVの位置づけとしてはETSI NFVのリファレンスプラットフォームのスピーディな開発で、基本的にコード開発は行わず、Requirementを書き、開発はOpenStack、ODL、OVS、KVMなどの各オープンソースコミュニティで行う方針。

OPNFVの位置づけ

- OPNFVは
  - NFVの要件を充足するシステムを
  - 新しいOSSを作るのではなく、既にあるOSSの組み合わせで実現する

  ことにフォーカスしている。
- このため、
  - ①既にあるOSSを選定し
  - ②動くことを確認し
  - ③機能不足は、各OSSにUpstream

  していくという行動原理になっている。

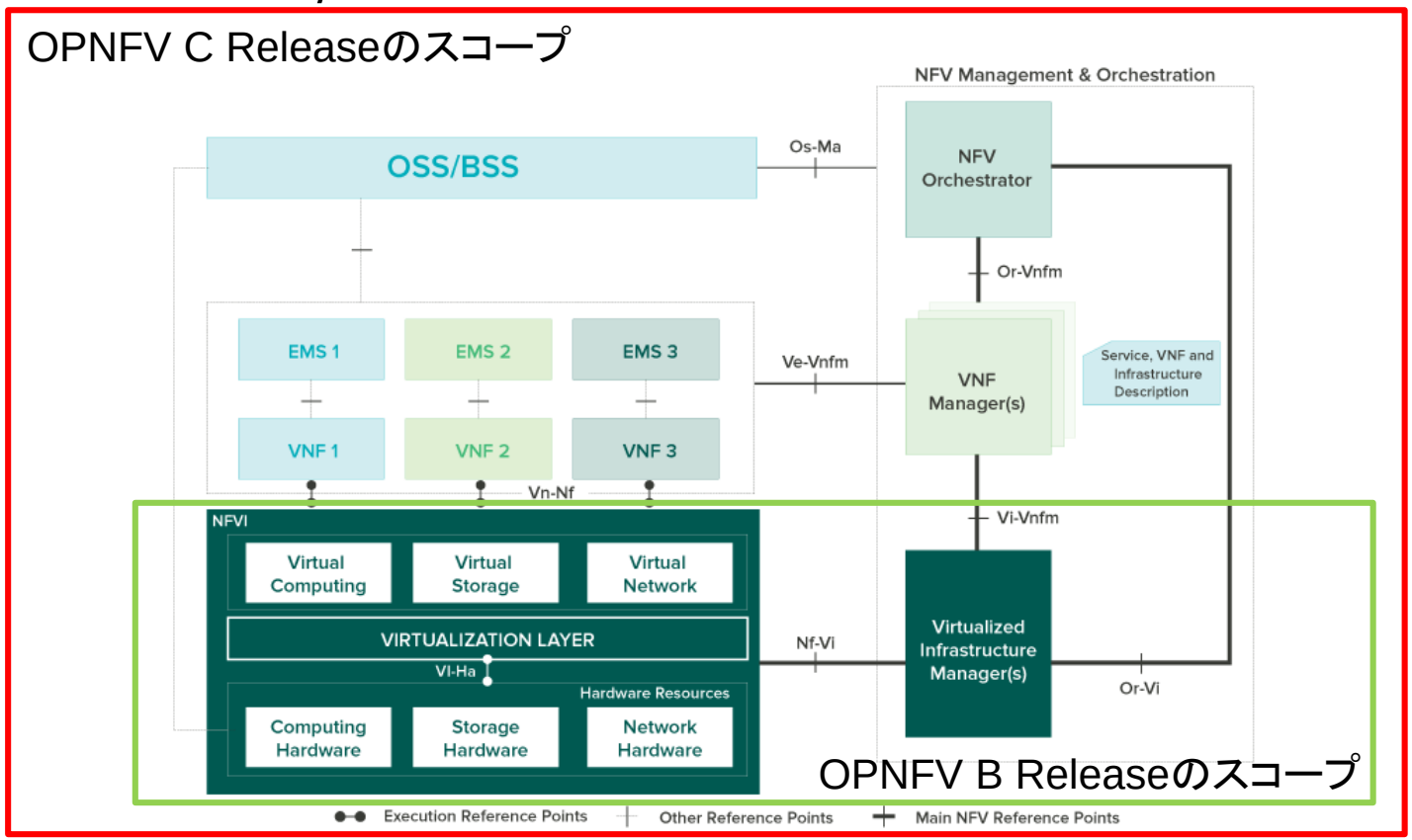


Upstreamに対する
開発や要件定義を行う。

Downstreamに対する
統合・デプロイ・テストを行う

OPNFVの位置づけ

- OPNFV Bリリースまでのスコープは VIM領域と限定していた。
- Cリリース以降はスコープの制限を無くし、ETSI NFVアーキテクチャ全体を含むようになった＝MANO、OSS/BSSも含むこととなった。



OPNFV B Releaseまで：VNFは非対象
OPNFV C Release以降；VNFMが入ってくるのでVNFが対象となる

オープンラボのOPNFVの取り組み

- Pharos Test Lab構築

- OOLとしてのOPNFV評価活動

  下記のテストプロジェクトを動作させて結果のまとめを行った

  - YardStick
  - Functest

- VNFテスト基盤のPoC開発
  - Functest上にVNFテスト基盤のPoCを開発
  - OPNFV Meetup Tokyo ＃1にて、開発したPoCを紹介

Pharos Test Lab構築

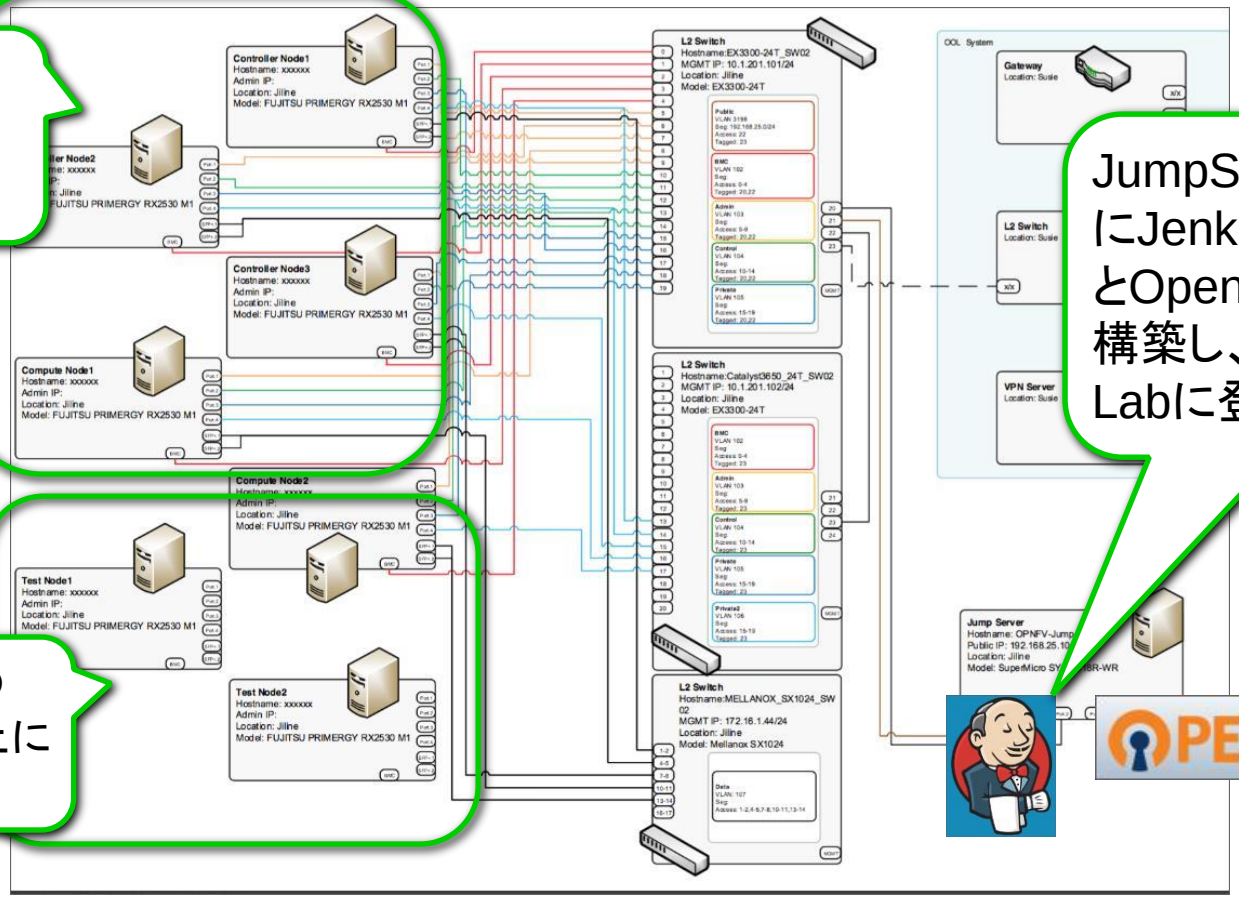
Pharos Test Lab構築

- OPNFVで定義している基本的な物理構成（POD）を用意
- 遠隔からの接続は、OPNFVで標準で使われているOpenVPNサーバを用意

1 Controller Node
3 Compute Node
でPODを構成

CI目的で貸し出し中のサーバ。各サーバー上にVirtual PODを構成

JumpServer上のVMにJenkinsのスレーブとOpenVPNサーバを構築し、Pharos Test Labに登録

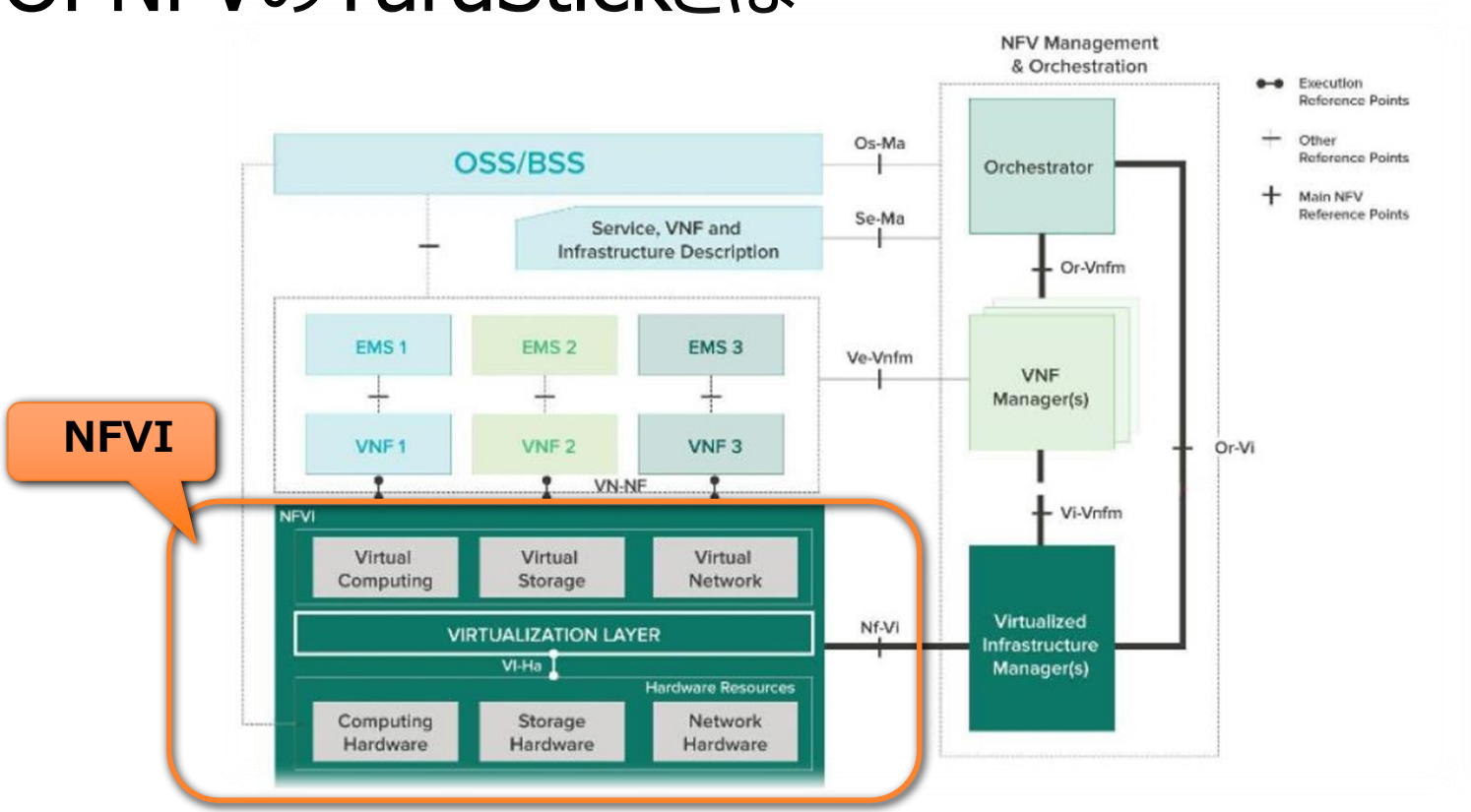


OOLとしてのOPNFV評価活動

- YardStick
- Functest

YardStickとは

・OPNFVのYardStickとは

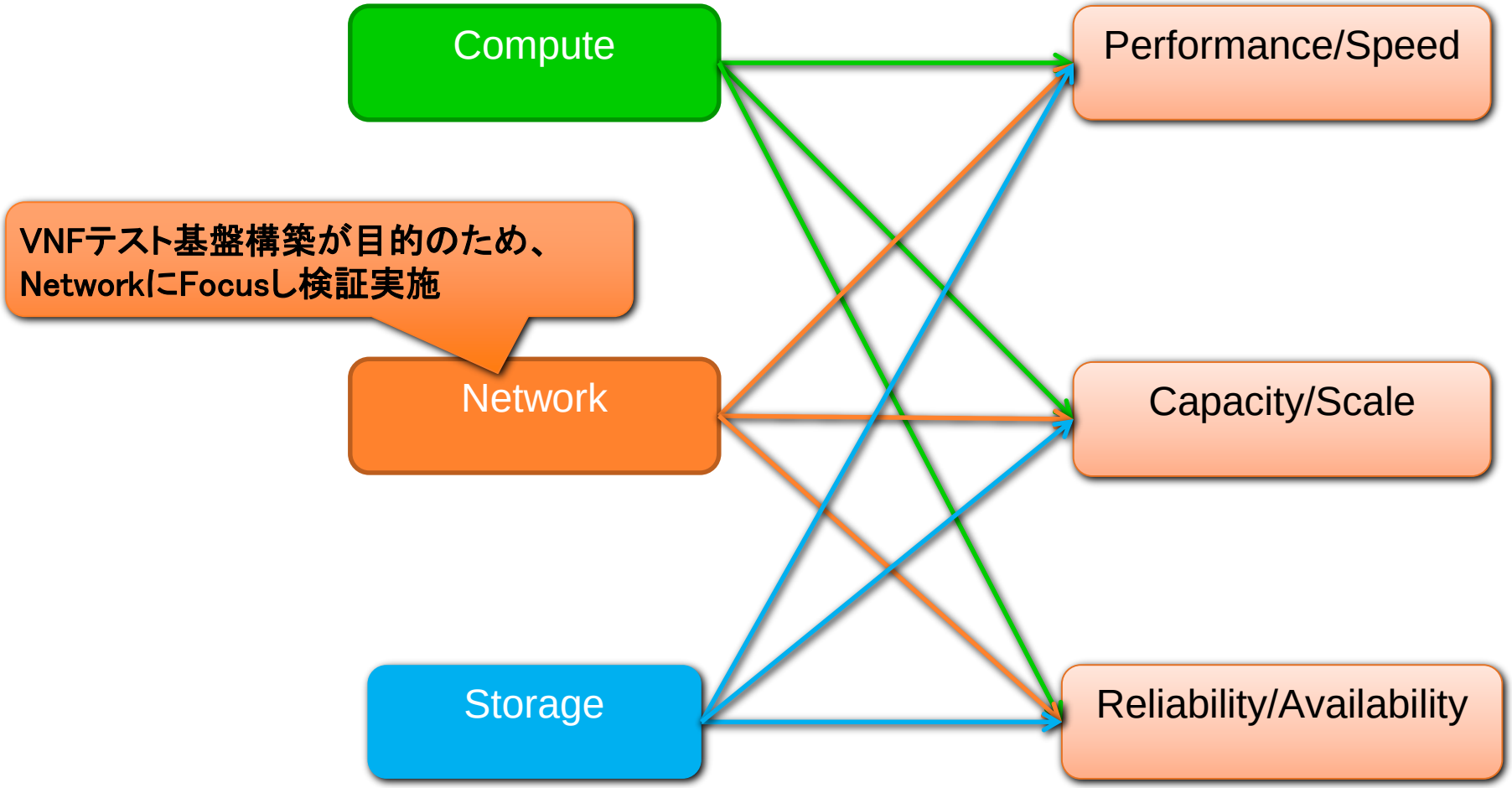


ETSI NFV アーキテクチャ

NFVIのテストするためのフレームワーク、
テストケースの開発をするProject

YardStickとは

- YardStickって何ができるの？



YardStickとは(Netowrk)

(1)Performance/Speed

スループット、Latency、ジッタ測定

- ・NFVIのノード間
- ・VM間

(2)Capacity/Scale

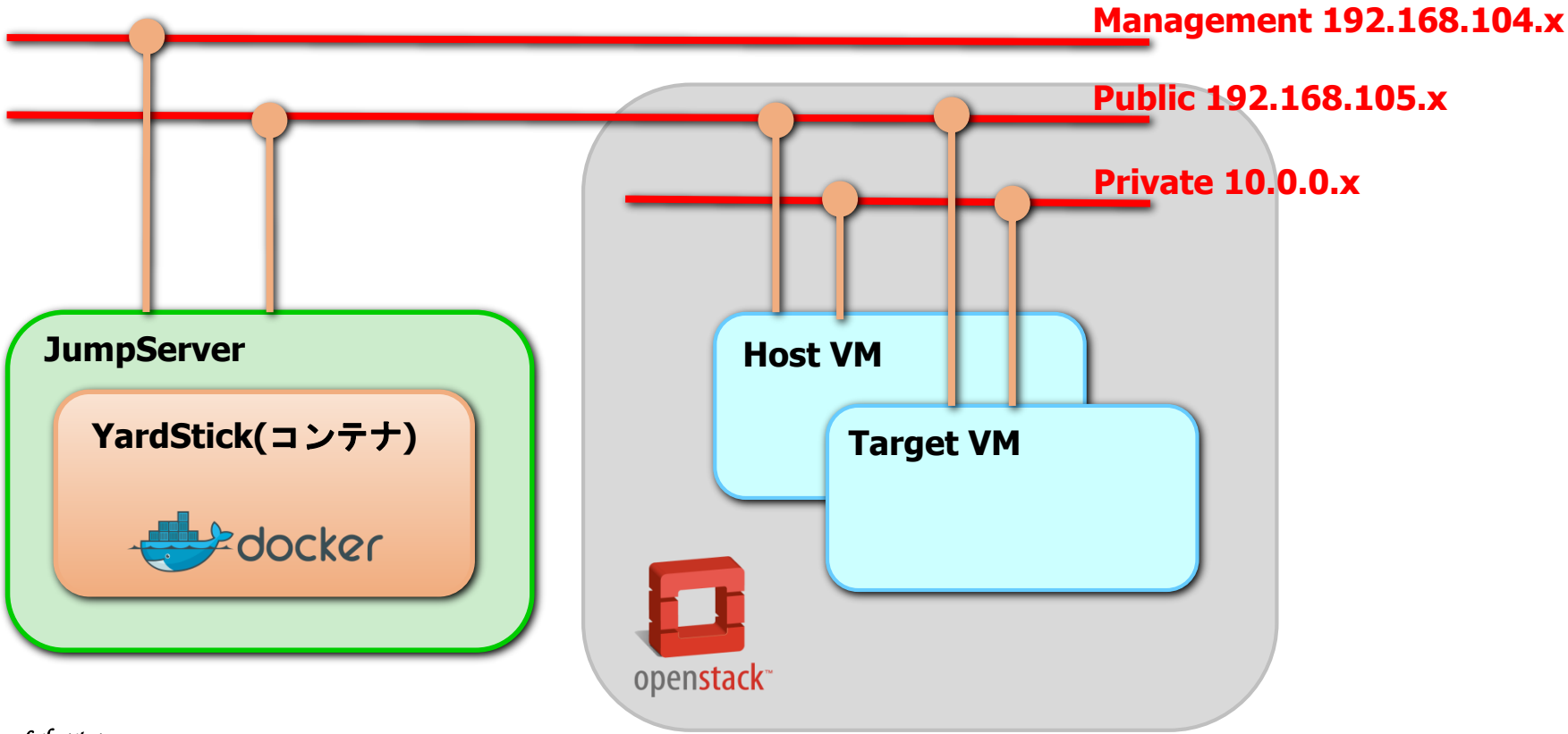
- ・接続数
- ・送受信フレーム数
- ・VM,NFVIのノード間の最大スループット

(3)Releavility/Availavility

- ・NICとLinkの可用性（接続時間）
- ・平均故障時間（MTTF）
- ・ Link切れるまでのネットワークタイムアウト時間
- ・フレーム欠損率

YardStick実行環境

- YardStickのDockerコンテナをJumpServerで起動させればOK。ネットワークはManagementとPublic、Privateネットワークを使用する。



YardStickテスト実行方法

- YardStickのテストケースは"yardstick task start <テストケース>"で実行できます。

```
root@:/# yardstick -d task start opnfv_yardstick_tc001.yaml
```

- -d ： デバッグログ出力

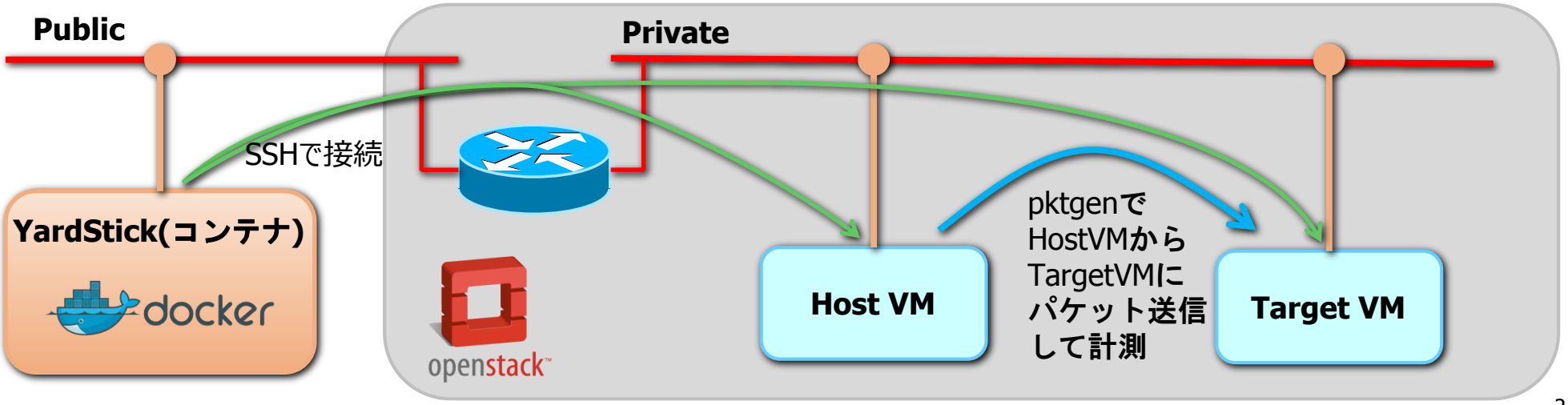
※試験結果は/tmp/yardstick.outに出力されます

試験実行の度に上書きされるため注意！

YardStickテストケースの一つ「TC001」の動作確認

YardStickテスト実行：TC001

- TC001試験内容
  - VM間でパケットを送信し、スループットを計測する
    - IaaSネットワークのパフォーマンス評価
- TC001試験構成
  - テストケース（YAML）で記載された構成をHeatでOpenStack環境上にPrivateネットワーク作成、VM作成、FloatingIP設定などを行い、SSHでVMに接続し、pktgenを使用して計測を行う。



YardStickテスト実行：TC001

- TC001の試験結果　※/tmp/yardstick.outにJSON形式で出力される


```
{
    "benchmark": {
        "data": {
            "packets_per_second": 629779,
            "packets_received": 12589740,
            "packets_sent": 12590832,
            "flows": 2,
            "errors": 0
        }
    }
}
```


計測結果（スループット）

約38.4Mbps

※629779pakets × 64bytes

YardStickの用途

- 幾つかのテストを動かしてみて感じたことは
  - OPNFV環境のNFVI（IaaS）の基本性能（ネットワーク性能など）を手間をかけずにテストすることができる。
    - VM間のスループットの計測
    - VM間のジッタ（受信パケットの遅延のばらつき、ゆらぎ ）の計測

    その他、ネットワークに限らず、VM自体の性能、Storageの性能も計測できそう。

  - YardStickのテストケースはYAML形式で記載されているため、テストシナリオのパラメータを変えることにより、比較的自由にテストケースを作成する事ができる。（TC001はポートのレンジ、パケットサイズ、測定時間、TC011は帯域幅が変更可能）

OOLとしてのOPNFV評価活動

- YardStick
- Functest

Functest

- Functestとは？

a.NFVI 上で、自動でネットワークとVMを作成してping確認

b.OpenStackのコンポーネントのTempestや、Rallyを使ったテスト

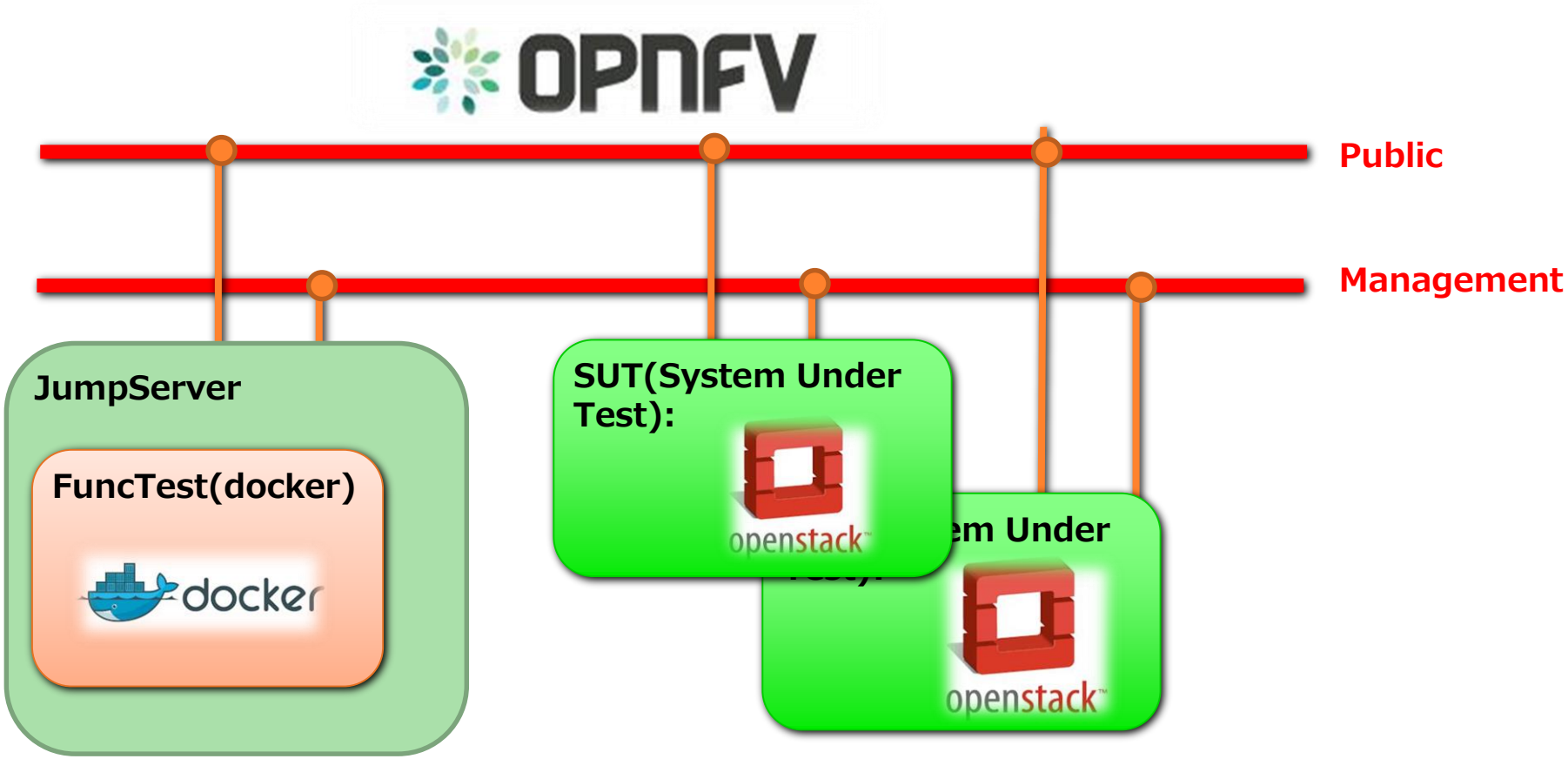
c.OpenDayLightのL2機能確認、ONOSのL2、L3機能確認

d.vIMS (IP Multimedia Subsystem：Qos制御や課金の機能を持つSystem) VNFをデプロイしてvIMS上でテストシナリオを流す。

e.Promise（リソース予約および管理プロジェクト）、Doctor（ネットワークサービスの障害管理とメンテナンス）のテスト

FUNCTEST実行環境

JumpServer上で動かせます。FUNCTESTのDockerコンテナをJumpServerで起動させればOK。ネットワークはPublicとManagementネットワークがつながっていればOK。



JumpServer： VPN経由で外からOPNFVで構築した環境にアクセスができるNode
FuncTest　： Dockerで動いていて、テストシナリオをキックするためのNode
SUT　　： テスト対象の環境
http://build.opnfv.org/artifacts.opnfv.org/functest/brahmaputra/docs/configguide/index.html

FUNCTESTのテストスイートの１つ「vPing_ssh」の動作確認

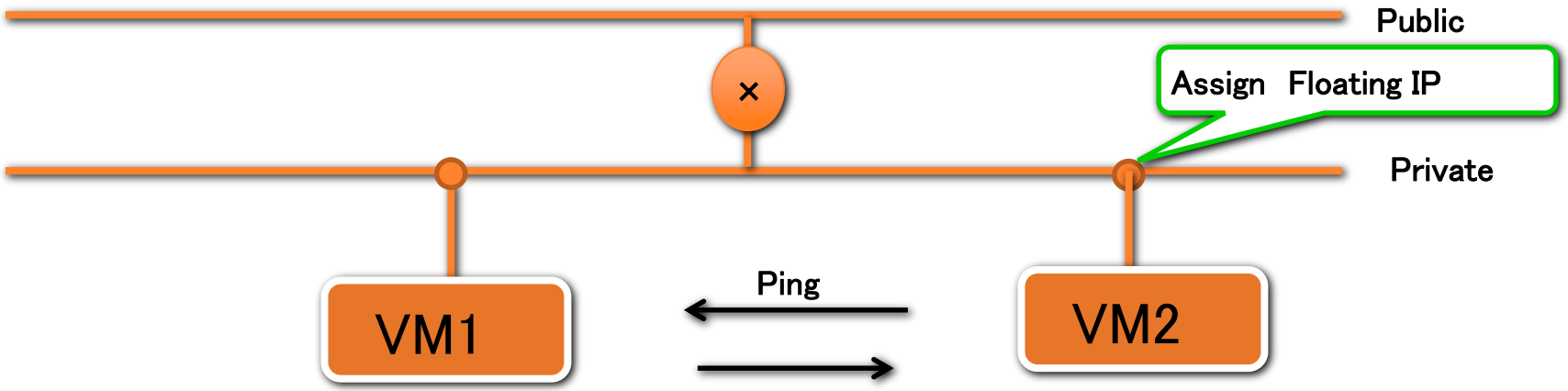
FUNCTESTの実行:vPing_sshとは

・vPing_sshとは

Functest のテストスイートの１つで、OPNFVで構築したOpenStack上にVMを２つ起動します。sshを用いてVM間の疎通確認をする。

・何のチェックをするの？

OPNFVで構築したOpenStack環境のネットワーク作成、VM作成、SSHでFloatingIP経由でVMにアクセス、疎通確認ができる。



FUNCTESTの実行:vPing_ssh

・vPing_sshの実行方法

docker上で run_tests.sh -t vping_ssh と実行する

※コマンドオプションで -t ｘｘｘｘの xxxx部分を変えるとTestSuiteが変更できる。例えば opendaylightの場合は、-t odl

```
root@f1b12fa12d39:~/repos/functest/docker# ./run_tests.sh -t vping_ssh↓
```

・vPing_sshの実行結果

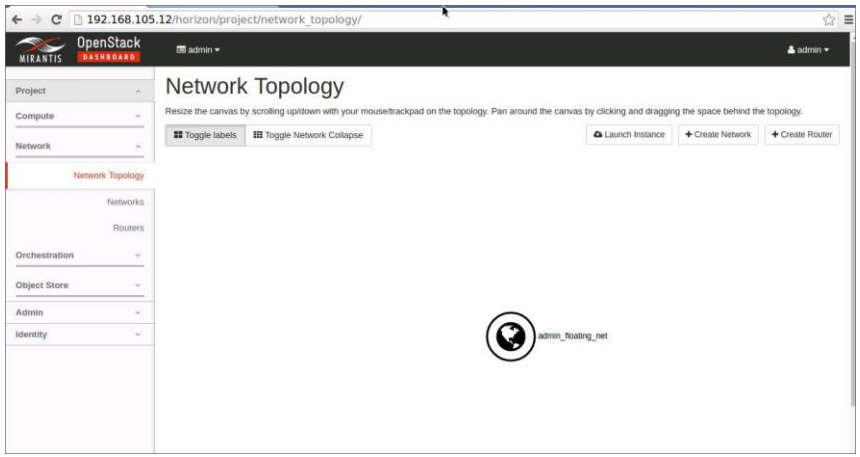
正常に成功すると最後に「vPing OK」と出力される

```
↓
2016-04-20 06:06:06,178 - vPing_ssh- INFO - vPing OK↓
[EOF]
```

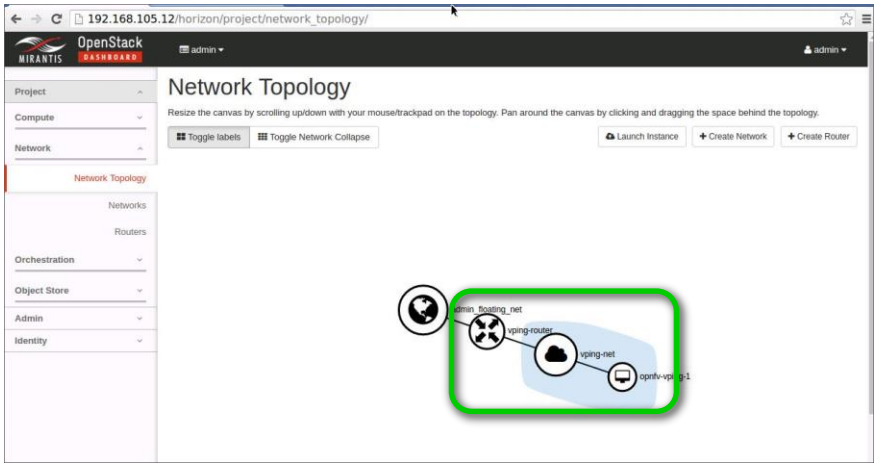
FUNCTESTの実行:vPing_ssh

vPing_ssh実行時のOpenStack DashBordでの確認：VMが起動が確認できる。

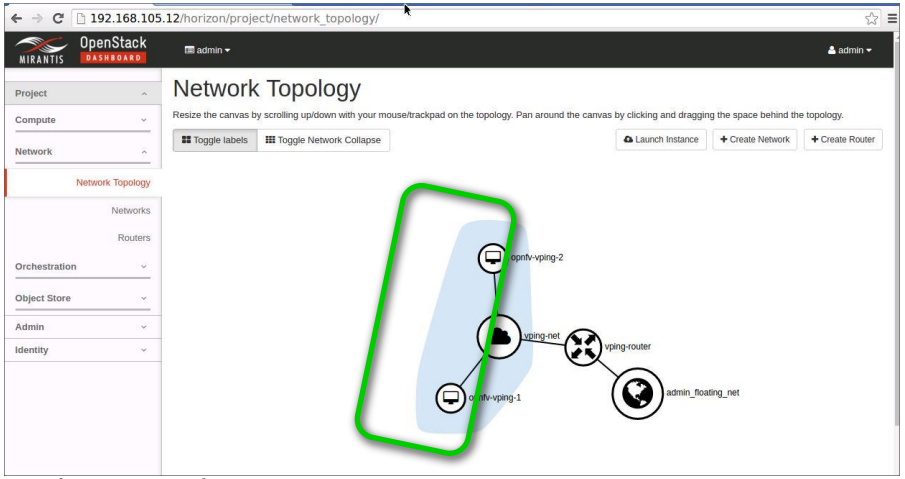
（1）外部ネットワークのみある



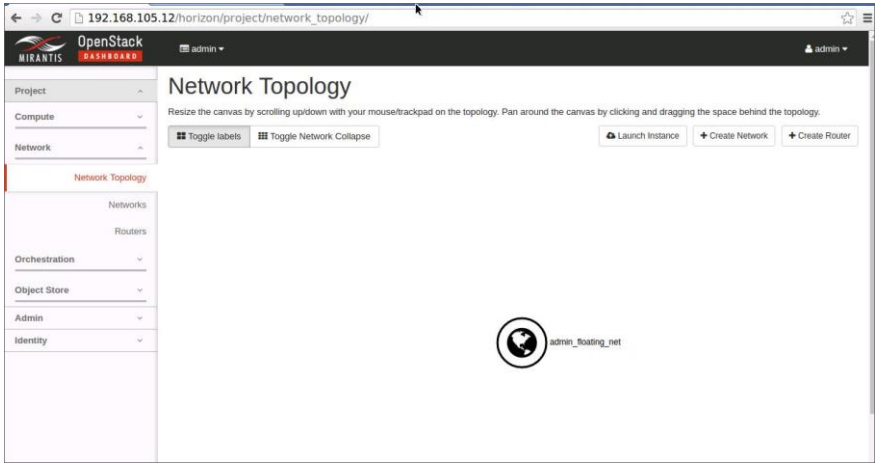
（2）外部ネットワークに接続するルータ、プライベートネットワークができる



（3）Ping確認用のVMが２つできて、疎通確認



（4）確認用のルータ、プライベートネットワーク、VMが削除される。



FUNCTESTの用途

vping_ssh,odlの二つを動かしてみて感じたことは

(1)OPNFV環境構築後、基本動作確認を手間をかけずにチェックができる。

a.ネットワーク、ルータ、VM、FloatingIP機能確認

b.OpenDayLight-OpenStack連携の機能確認

(2)FUNCTESTのテストシナリオを拡張することでOPNFV上に構築したシステムのテスト自動化ができると考える。

前提条件： openstack client、REST APIを使用可能なこと。

VNFテスト基盤のPoC開発

オープンラボのNFVの取り組み

- 背景

  OOLでは、「クラウド」と「SDN」の融合した技術としてNFVの検証に取り組んでいる。昨年度(2015年度)は 、VNFのテスト自動化の検証を実施

- PJ名

  VNFテスト自動化プロジェクト

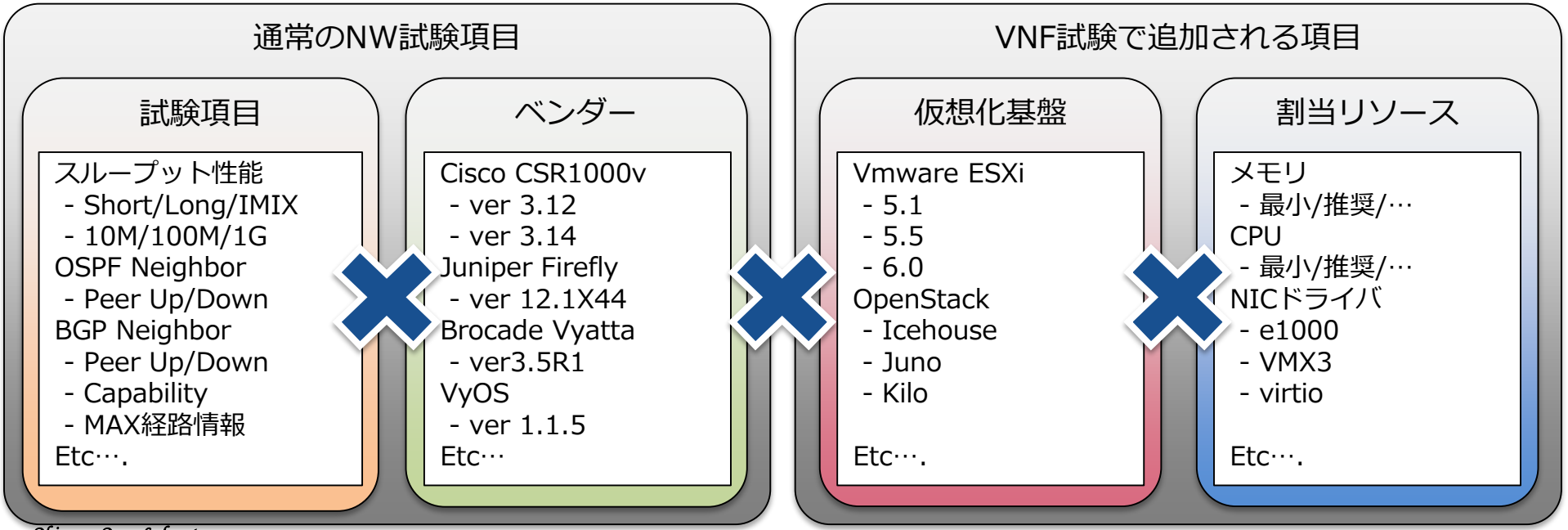
- 研究目的

  VNFアプライアンスのネットワーク試験自動化の検討、及びPoC構築

- 検証内容
  - 機能試験
  - 相互接続試験
  - 性能試験

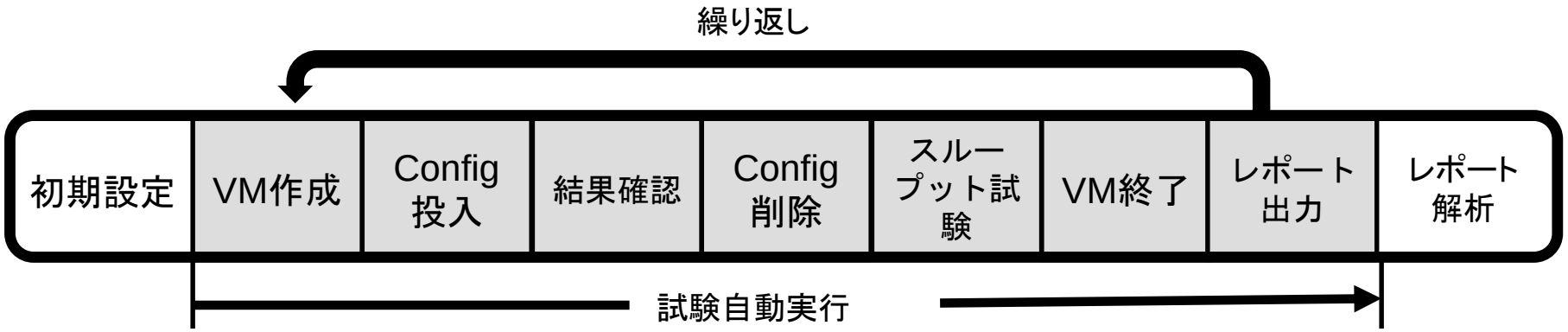
VNFテスト自動化PJの取り組みの背景

- 仮想アプライアンス市場が盛り上がってきている
  - Router, Switch, Firewall, etc...
    - 普及の過程で導入例が多くはない（どの程度つかえるのか？）
    - 検証の知見が少ない（どんな検証をすればよいのか？）
- 試験項目数の爆発的増加
  - VNF試験特有の試験項目パターンが追加
    - 手作業での実施は困難



VNFテスト自動化PJの取り組みの成果

- 試験自動化による成果
  - 必要な条件の試験を実行し，結果出力までの自動化を実現



- 試験自動化による恩恵
  - 人的・時間的コストの削減
  - ヒューマンエラーのリスクを低下
  - 検証対象の仮想アプライアンスの知識・技術を持たない作業者でも試験実行することが可能

VNFテスト自動化PJの取り組みの成果

- 手動で実施することが困難な網羅的試験の自動化を実現

試験対象ルータ	【CSR1000v,Firefly,Vyatta,VyOS】
試験対象ルータ用仮想化基盤	【VMware ESXi, OpenStack】
試験対象ルータハードウェア構成	【CPU,Mem,HDD,VNIC】
リファレンスルータ	【CSR1000v,Firefly,Vyatta,VyOS】
リファレンスルータ用仮想化基盤	【VMware ESXi, OpenStack】
確認項目	【BGP:5種,OSPF:7種】
トラフィック試験	【パターン:3種,サイズ:6種】

2736通りの組み合わせ試験を自動実施可能
（約40時間）

VNFテスト自動化をOPNFVで実現したい

VNFテスト自動化をオープンソースで実現

- なぜOPNFVか？
  - NFV向けプラットフォームの存在感

  - OPNFVはオープンなNFV向けのプラットフォームとして、存在感を増していると考える。
  - 予想の域を出ないが、OpenStackとの関係が強化されていくのであれば、デファクトスタンダードとなっていく可能性は高いと考える。

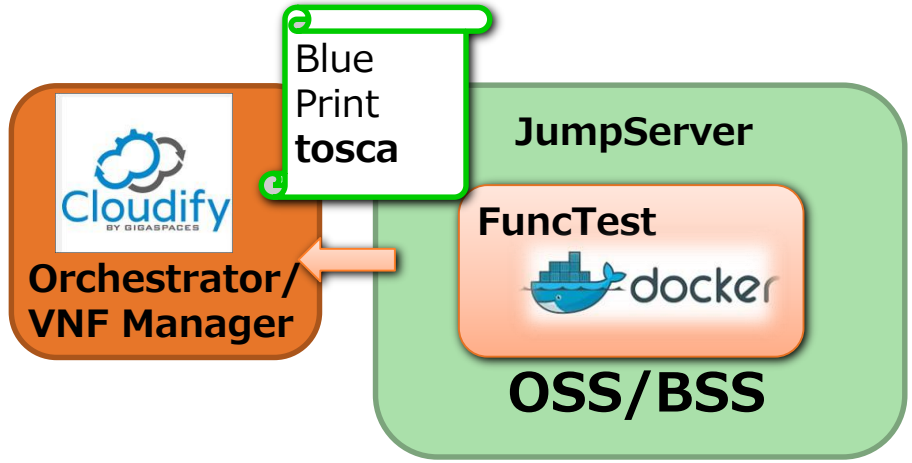
VNFテスト自動化をOPNFVのFunctestで実現したい

- ・なぜFunctest？

FuncTestはOPNFVのテストProjectの一つで
オープンラボのやりたいことと方向性が一致

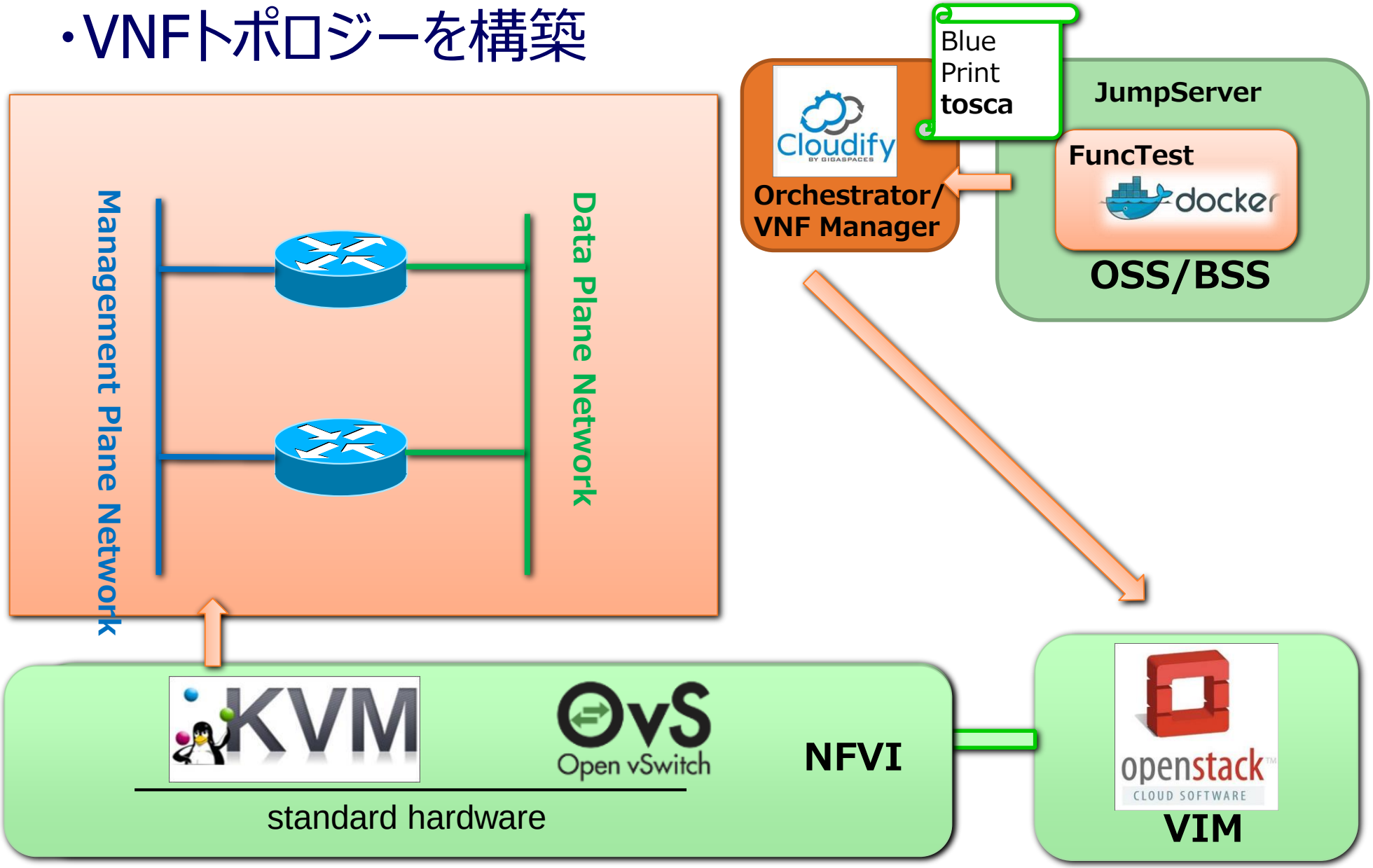
VNFテスト自動化の検討

・VNFトポロジーを構築



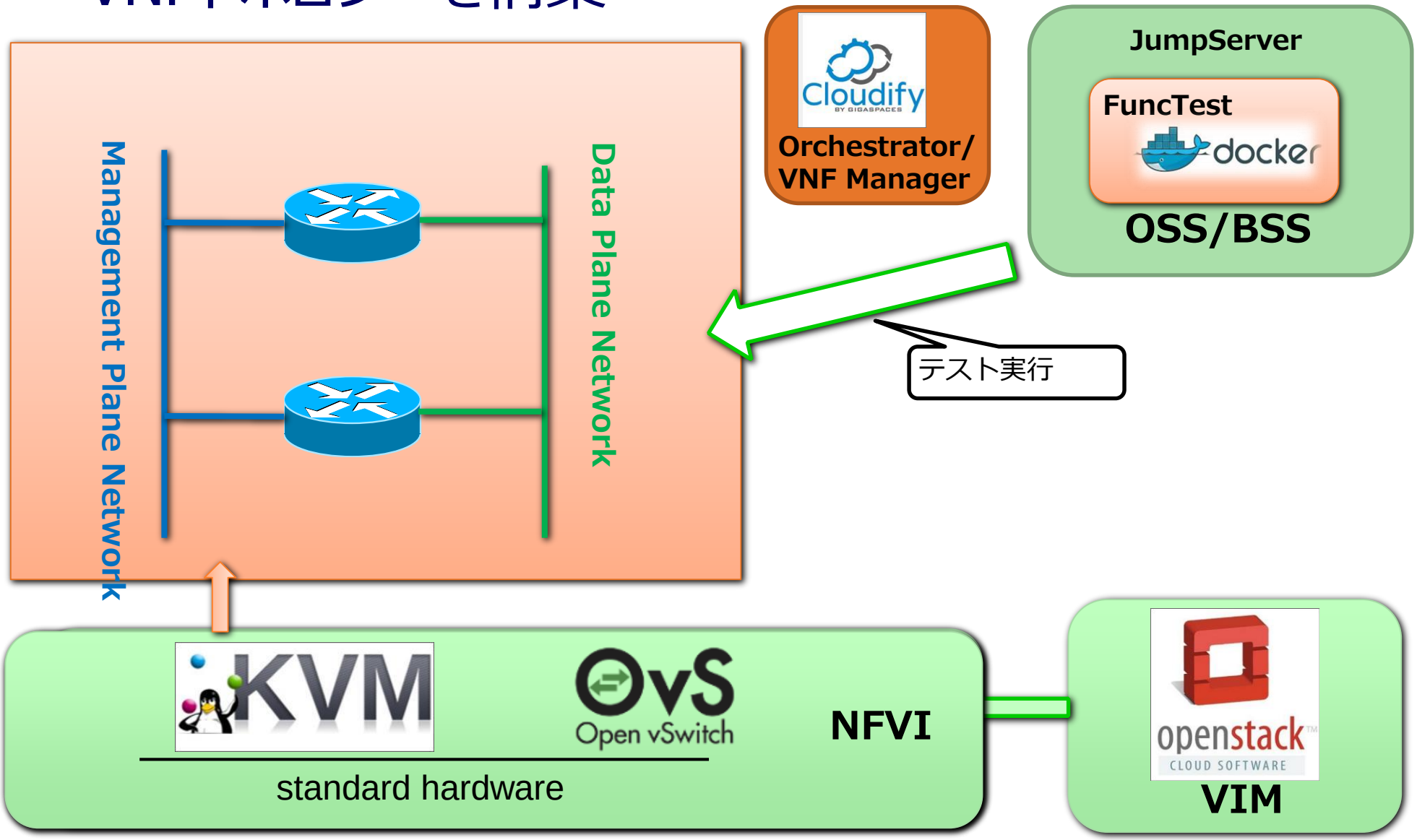
VNFテスト自動化の検討

・VNFトポロジーを構築



VNFテスト自動化の検討

・VNFトポロジーを構築



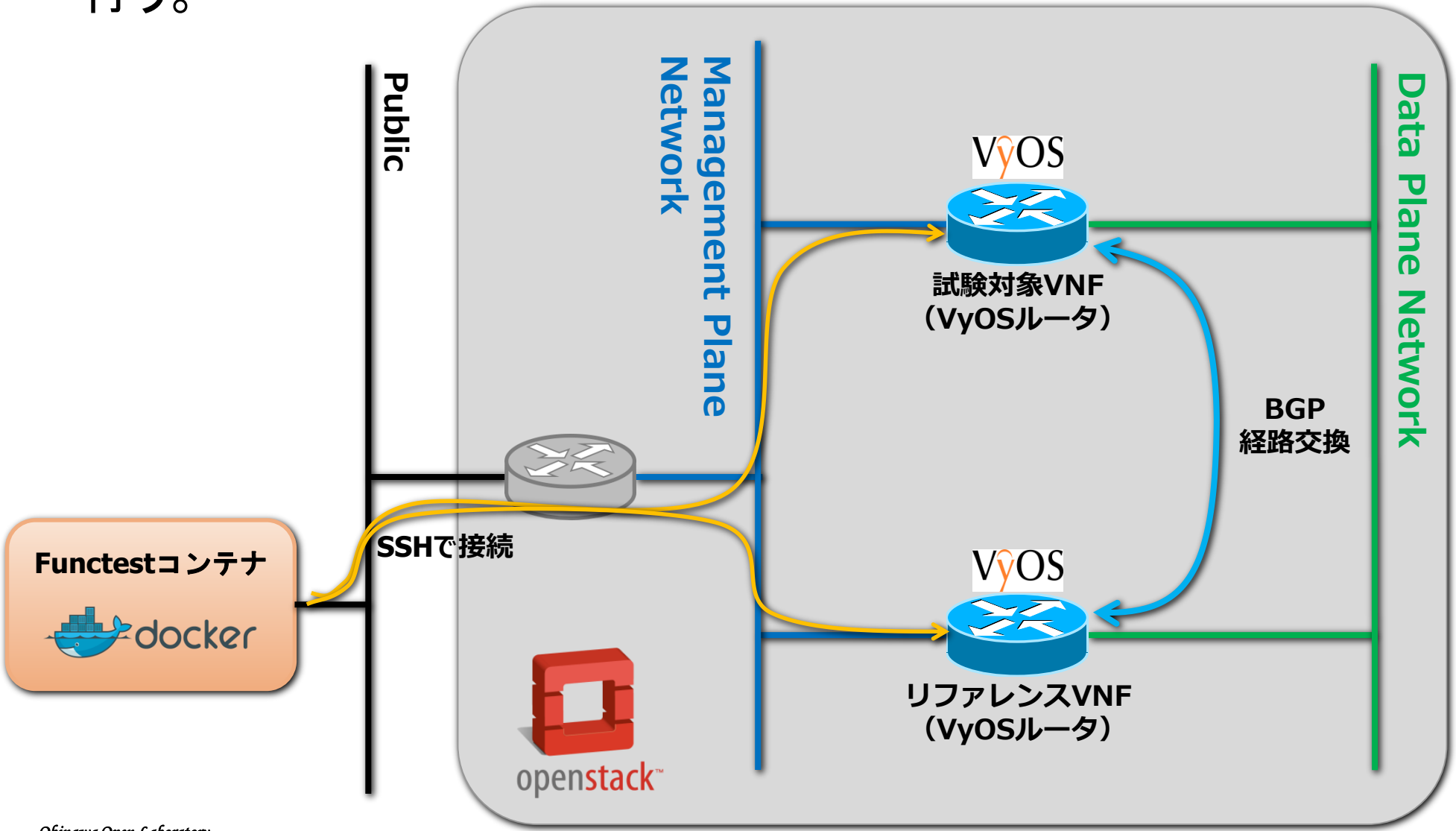
実装して動かしてみた VNFテスト内容

- まずは、スモールスタートでBGPの経路交換の相互接続性試験の自動化をFuncTestをベースで実装してみました。
  - 試験構成

  - 試験対応のVNF　：VyOS (ver1.1.7)
  - 試験項目　　:相互接続性試験: BGP
    - 確認内容
      - BGPの接続状態
      - 経路広告数
      - 経路情報が正しく送出されているか
      - 経路情報が正しく受信しているか
      - ルーティングテーブルが正しいか

試験構成

- 試験対象ルータ、リファレンスルータ間で相互接続性試験を行う。

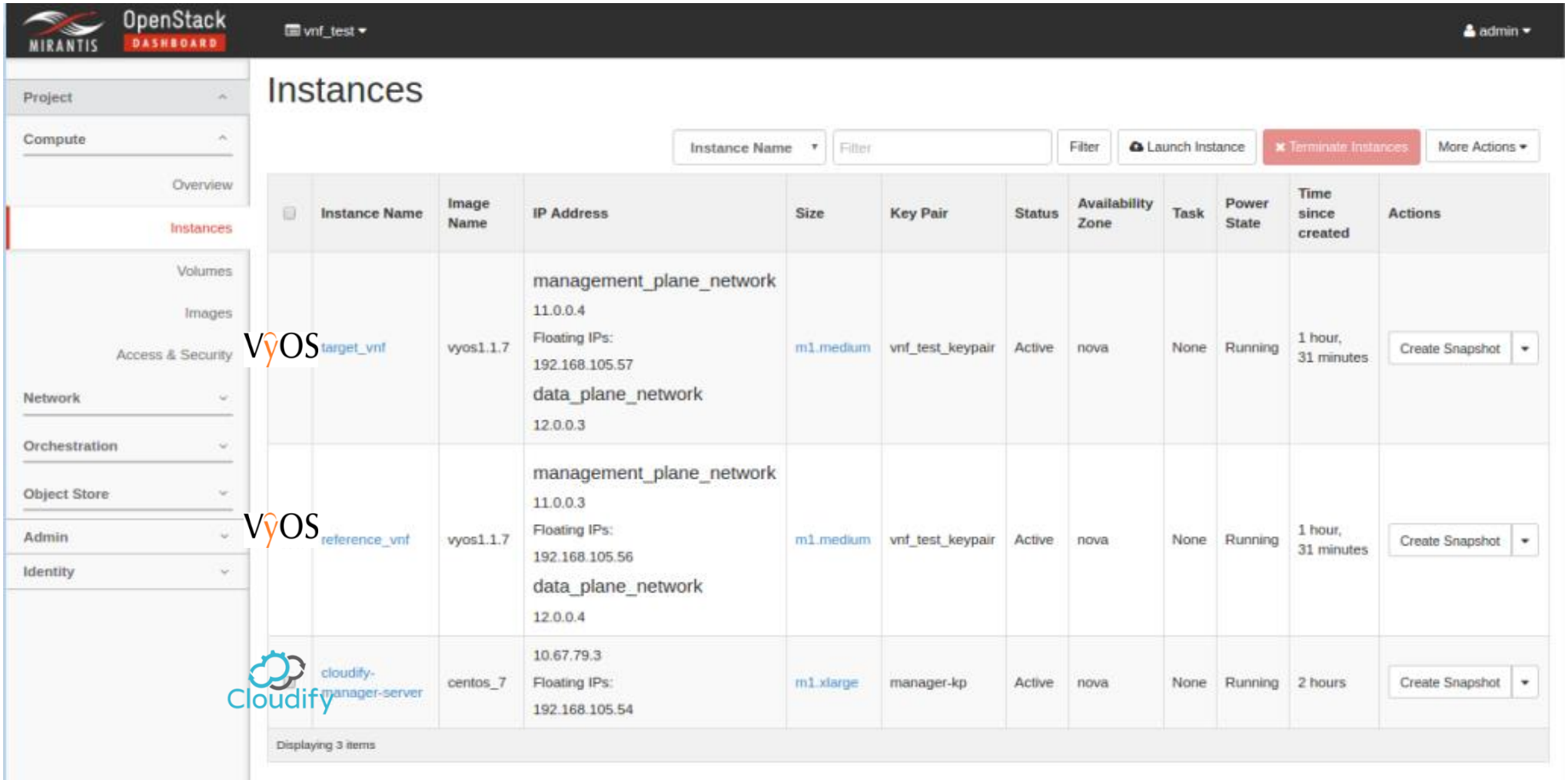


試験実行方法

- VNFテストの実行方法
  - Functestのコンテナ上で"functest testcase run vnf_test"で実行します。

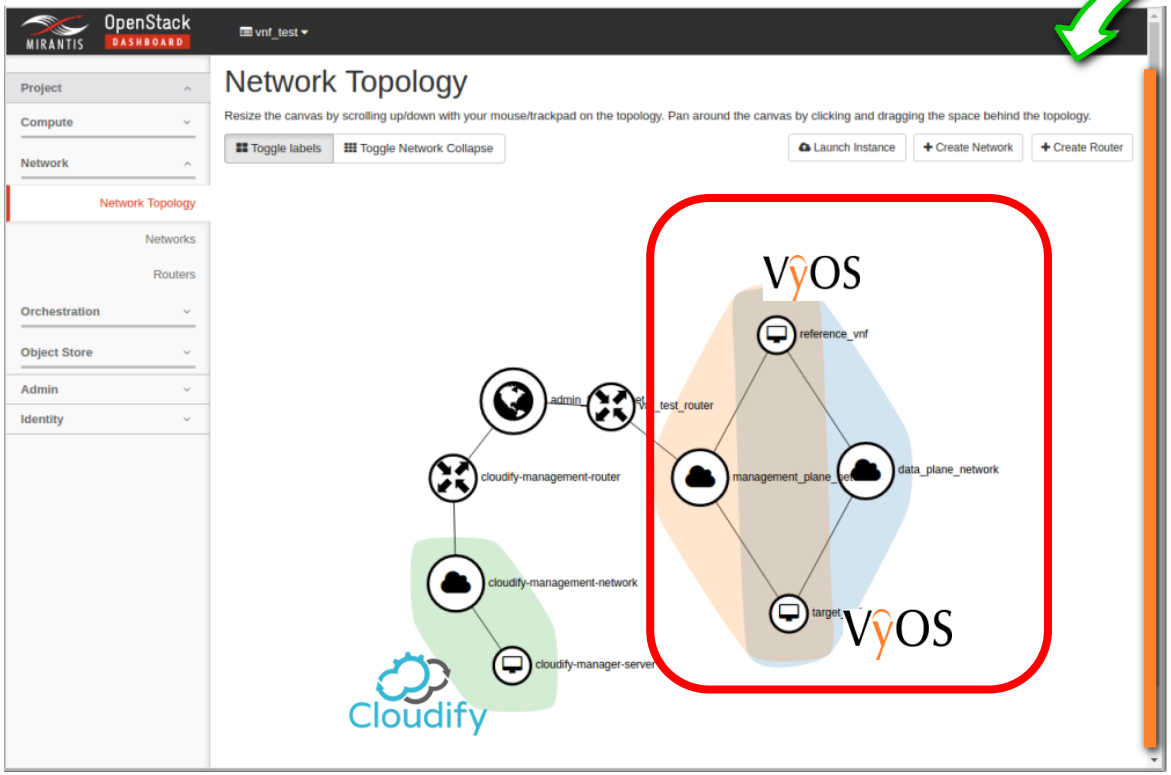
```
root@ceaeea9a5272:~# functest testcase run vnf_test↓
```

試験結果（OpenStack上のインスタンス）

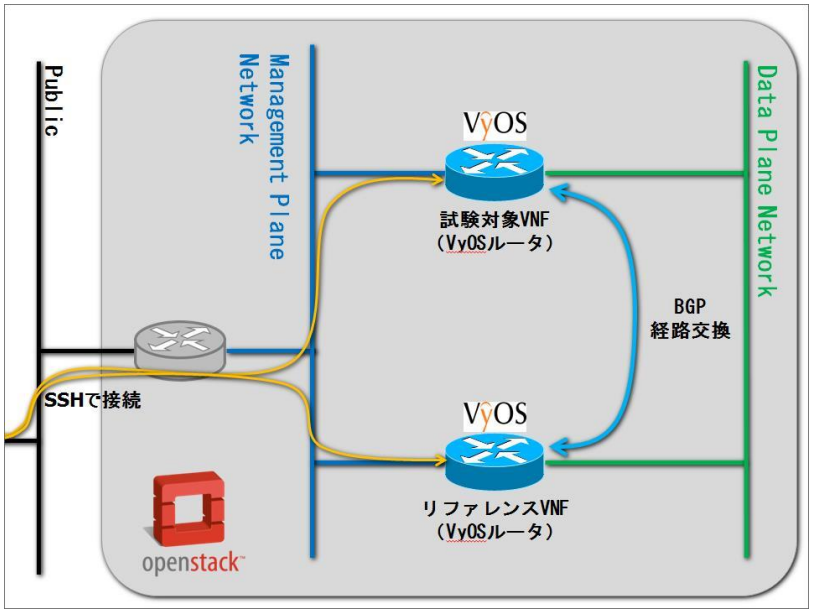


試験結果（OpenStack上のネットワーク）

- OpenStack上のネットワークトポロジ



試験トポロジー



OpenStack上にVNFテストのネットワークトポロジがデプロイされ
テストを実施します

試験結果

- VNFテストの実行結果
  - 各テスト項目について結果を表示します。

```
vnf_test.ssh_client - INFO - SSH connect to 192.168.105.129.
vnf_test.ssh_client - INFO - SSH connection established to 192.168.105.129.
vnf_test.cecker - INFO - ============================================================
vnf_test.cecker - INFO -  Check bgp peer                                        OK |
vnf_test.cecker - INFO - ============================================================
vnf_test.cecker - INFO -  Check bgp status                                      OK |
vnf_test.cecker - INFO - ============================================================
vnf_test.cecker - INFO -  Check route advertise                                 OK |
vnf_test.cecker - INFO - ============================================================
vnf_test.cecker - INFO -  Check route receive                                   OK |
vnf_test.cecker - INFO - ============================================================
vnf_test.cecker - INFO -  Check route table                                     OK |
run_tests - INFO - Test execution time: 04:04
run_tests - INFO - Execution exit value: 0
```

経路広告数確認

BGPの接続確認

経路情報送出確認

経路情報受信確認

Routing Tableの確認


Thank You!
ご清聴ありがとうございました