Vol.2019-IOT-44 No.15
2019/3/7

SSH Kernel: Jupyter Notebookでサーバの遠隔運用手順を記述・実行するJupyter拡張の開発

上野 優[1,a)]　今井 祐二[1,b)]

概要：サーバ運用の手順書を Jupyter Notebook を用いて記述し，作業内容・コマンド・実行結果を一つの「実行可能な手順書」にまとめることで，作業の省力化と同時に，運用ノウハウの共有を円滑にするアイディアが提案されている．しかし，遠隔サーバを運用対象とする場合，従来の対話型端末と比べコマンドの記述形式を大きく変える必要があり，適用の妨げとなっていた．そこで筆者らは，この問題を解消する Jupyter 拡張である「SSH Kernel」を開発し，トライアルユーザへのアンケートにより提案手法の有効性を確かめた．

SSH Kernel: An implementation of Jupyter extension to improve the usability of remote server administration on Jupyter Notebook

1. はじめに

近年，システム基盤の運用作業を高い再現性で実施するために，作業手順の説明文書とコマンド操作を Jupyter Notebook で記述する方法論とそれを支援するツール群として「Literate Computing for Reproducible Infrastructure」[1] [2] [3] や「Executable Runbook」 [4] が提案されている．これらは，ドキュメントとコードを同じテキストで一元管理する「Literate Computing」の考え方を 「Jupyter Notebook」 [5] を用いてコンピュータのコマンド操作に適用するものである．本稿ではこれらの方法論に従って記述されたテキストを指して「実行可能な手順書」と呼ぶ．

システム基盤を構成するサーバがオペレータから地理的に離れた場所にある場合，オペレータは SSH 端末エミュレータを用いて遠隔サーバにログイン後，「端末にコマンドを入力し，実行し，結果を確認し，その結果に応じて次のコマンドを入力する」というイテレーションステップを繰り返し，運用作業を行う．従来，この作業を適確に行うため，イテレーションステップは事前に手順書に記述され，オペレータは手順書に記述されたコマンドをコピーペーストで端末に入力している．本稿ではこの手順書を「従来型手順書」と呼ぶ．

「従来型手順書」を「実行可能な手順書」に書き換えてサーバの遠隔運用に適用しようとすると，「コマンド入力，実行，結果確認」のイテレーションステップの度に SSH セッションを張り直すつくりになる．この場合，直前のコマンド実行後の状態が失われるため，次のイテレーションステップでわざわざ状態を戻す必要があった．そのため，従来のコマンド操作をそのまま「実行可能な手順書」に記述することができなかった．

そこで筆者らは，イテレーションステップをまたいで実行コンテクストを引き継ぐことを特長とする，Jupyter Notebook 拡張である「SSH Kernel」を開発しオープンソースとして公開した． SSH Kernel により，遠隔サーバ運用への「実行可能な手順書」の適用が容易になり，作業の定義やオペレータ間のノウハウ共有をより円滑にできる．

本稿の構成は次の通り． 2 節では Jupyter Notebook を用いた「実行可能な手順書」について述べ，3 節では「実行可能な手順書」をサーバの遠隔運用に適用する際の課題を述べる．4 節で SSH Kernel の設計と実装を述べ，5 節で評価を行う．

2. Jupyter Notebookを用いた「実行可能な手順書」

本節では，「実行可能な手順書」の概要と，その実現形態

[1] 株式会社富士通研究所
FUJITSU LABORATORIES LTD.
[a)] ueno.masaru@fujitsu.com
[b)] yuji.imai@fujitsu.com

を説明する．

2.1 「実行可能な手順書」の概要

システム基盤の運用作業は，「作業の目的，コマンドの意味，コマンド列，結果の確認方法」が記述された文書に基づいて行われる．作業をあらかじめ文書とすることで，操作の正確性が向上し，オペレータ間の引継ぎが円滑になることが期待できる．本稿ではこの文書を「従来型手順書」と呼ぶ．「従来型手順書」はオペレータが解釈する前提のドキュメントであり，オペレータは記述されたコマンドを端末にコピーペーストして実行する必要がある．これにより実行コストが増え，入力ミスが発生するリスクが生じている．

一方，Ansible [*1] や Chef [*2] やシェルなどによる自動化スクリプトによるサーバの運用が広まっている．自動化スクリプトは，機械的に実行することができ，端末へコピーペーストしなくても簡単に実行することができるため，実行コストを低下させ，入力ミスのリスクも排除できる．しかし，オペレータ向けの文書としては，コメントによる補足的な記述しかできないため，「従来型手順書」に比べて文章の可読性が低くなり，引き継ぎの円滑さに劣る．

近年，システム基盤の運用作業を高い再現性で実施するために，「実行可能な手順書」のアイディアが提案されている．これは，コマンド列を機械が人手を介さず「実行可能」であると同時に，「従来型手順書」と同様オペレータが解釈しやすい「手順書」の体裁がとれることを特長とする．文献 [1] の「Literate Computing for Reproducible Infrastructure」や [4] の「Executable Runbook」は，実現手段として Jupyter Notebook を利用する．

図 1 に，Jupyter Notebook を用いた「実行可能な手順書」のノートブックユーザインタフェースを示す．1 つのノートブックは複数のドキュメント部分とコード部分とコード実行結果部分から構成される．「従来型手順書」に記載されたコマンドをコード部分に記入し選択された部分を実行するボタンを押すと，コマンド実行され実行結果が記録される．

2.2 Jupyter Notebook のアーキテクチャ

「実行可能な手順書」の実現手段である，Jupyter Notebook のアーキテクチャを図 2に示す．

- Jupyter Notebook (①) は，ユーザがノートブックファイル (②) を編集・実行するのに用いる，Web アプリケーションである
- ノートブックファイルは，ドキュメント部分とコード部分，コードの出力部分を一元管理するファイルである

*1 https://www.ansible.com/
*2 https://chef.io/

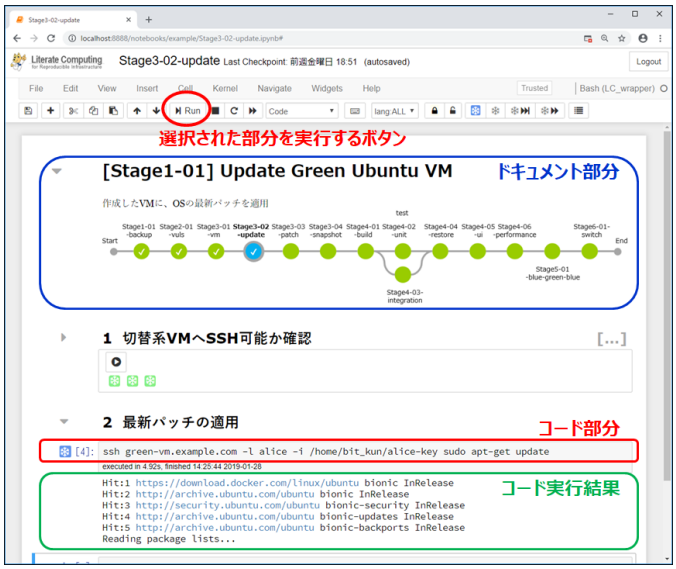

図 1 Jupyter Notebook を用いた「実行可能な手順書」のノートブックユーザインタフェース

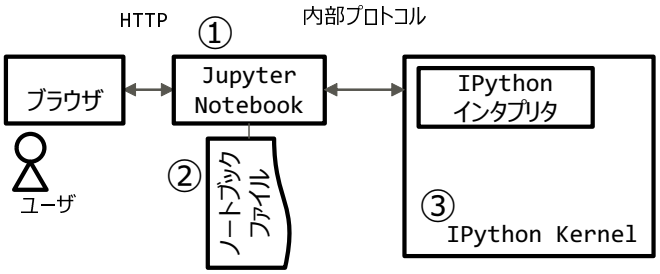

図 2 Jupyter Notebook のアーキテクチャ

- Jupyter Notebook のバックエンドで動作し，ノートブックファイルのコード部分を実行する拡張プログラムを，Jupyter 開発者コミュニティでは「カーネル」と呼ぶ．多数のプログラミング言語をサポートするために，言語ごとのカーネルが開発・公開されている
- IPython Kernel (③) は，与えられたコード部分を Python コードとして実行する Jupyter Notebook 標準のカーネルである．Python コードだけでなく OS のコマンドを実行する機能（外部コマンド実行機能）を備え，OS のコマンドをノートブックに記述・実行できる

3. サーバの遠隔運用に「実行可能な手順書」を適用する際の課題

本節では，「従来型手順書」を前提としたサーバの遠隔運用に「実行可能な手順書」を適用するにあたって，コード部分を遠隔実行するためのモジュール配置を 2 種類述べ，次いでそれぞれの配置について適用の課題を述べる．

3.1 モジュール配置

「実行可能な手順書」のコード部分を遠隔実行するためのモジュール配置は，次の 2 通りが候補となる．

配置 A 「カーネル」が対象サーバで動作する配置で，コマンドは Jupyter Notebook からカーネルに内部プロトコルで受け渡され遠隔実行される（図 3）．

配置 B SSH デーモンが対象サーバで動作する配置で，コマンドは SSH 経由で遠隔実行される（図 4）．配置 A と異なり，カーネルは対象サーバではなく Jupyter Notebook と同じサーバで動作する．

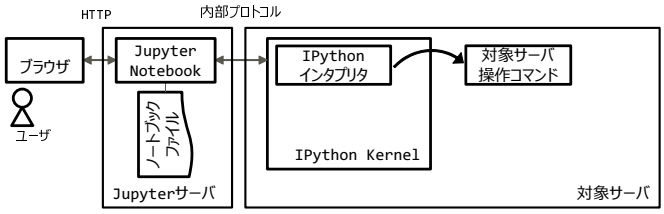

図 3 配置 A. カーネルが対象サーバで動作する

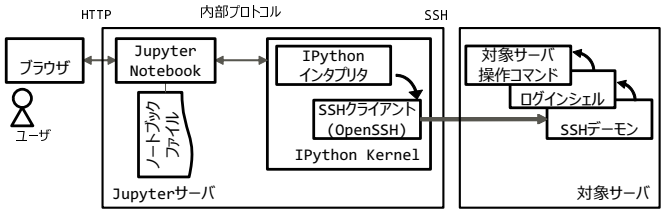

図 4 配置 B. SSH デーモンが対象サーバで動作する

3.2 運用対象へのソフトウェア導入・維持の課題

配置 A では，対象サーバにカーネルを導入する必要がある．しかし実運用の現場では，対象サーバへのソフトウェア導入が運用業務の契約上難しいケースや，対象サーバ数が多くそれぞれにカーネルを導入し維持するコストが許容できないケースがあり，配置 A にすることが現実的でない．

3.3 外部コマンドとして遠隔操作を行う際の課題

配置 B では，IPython Kernel の「外部コマンド実行機能」を用いてノートブックに「遠隔実行するコマンドを引数とする”ssh”コマンド」を記述する必要がある．図 5に「従来型手順書」のコマンド操作を，外部コマンド実行機能を用いて「実行可能な手順書」に書き換える例を示す．

「実行可能な手順書」の記述は「従来型手順書」の記述に比べ長く複雑になっている．特に以下の点が記述者の負担であるとともに可読性も大きく損なう要因になっている．

- SSH セッションを毎回張り直すことで直前のコマンド実行後の状態が失われるため，それを復元するための環境変数やカレントディレクトリを毎回与える必要がある
- 遠隔実行するコマンドに含まれるシェルのメタ文字（改行，バッククオート，$変数など）を，IPython Kenrel の外部コマンド実行機能に評価されないよう適切にエスケープ文字を加えながら，イテレーションステップ内のコマンド全てを外部コマンド実行に書き換える必要がある

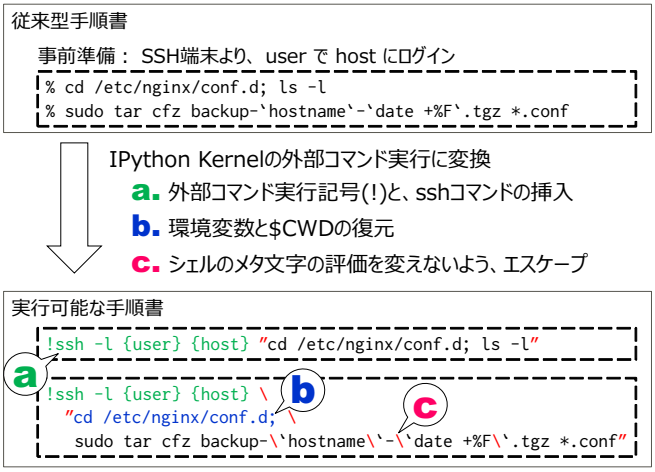

図 5 「従来型手順書」のコマンド操作の「実行可能な手順書」への書き換え例

4. 提案手法

前節の課題を解消するため次のように要件を定義した．

- 対象サーバへのソフトウェア導入が不要
- 「従来型手順書」記載のコマンドを，書き換えることなくそのまま記述可能
- 遠隔コマンド実行の度に，コマンド実行後の状態を失わず，次のコマンド実行前に状態を復元するための書き足しが不要

これらの要件を満たすため，与えられたコード部分を遠隔コマンドと見なし，実行コンテクストを引き継ぎながら実行するカーネルである「SSH Kernel」を提案する．

図 6に，提案手法の構成図を示す．このように前節の「配置 B」をとるため，対象サーバへソフトウェア（カーネル）を導入する必要は無い．

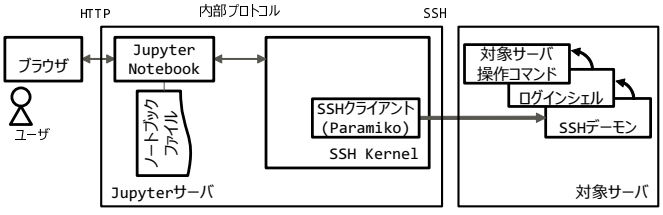

図 6 構成図

4.1 SSH Kernel の概要

ここでは，開発した SSH Kernel の概要を示す．

図 7 は，SSH Kernel と Jupyter Notebook を用いた「実行可能な手順書」の記述・実行例である．

①「%login 〈ホスト名〉」「%logout」は，対象サーバへの SSH セッションを開設・切断するための SSH Kernel 独自の命令である．ログイン時に必要な，接続先と認証情報の組（ホスト名・IP アドレス，ユーザ名，秘密鍵）は OpenSSH クライアントの設定ファイル (~/.ssh/config ファイル) を参照する．

②コード部分は，「%login〈ホスト名〉」命令に与える対象サーバ上の OS コマンドとして遠隔実行される．このコードは外部コマンド実行機能を経由せず，直接対象サーバのログインシェルに渡されるため，シェルメタ文字のエスケープは不要である．コードの繰り返しの実行を通じて，カレントディレクトリと環境変数は「従来型手順書」と同様に引き継がれる．

③ノートブックのコード実行結果部分には，そのコード（コマンド）をどのサーバで，誰が，どの作業ディレクトリで実行したのかを明らかに提示できるよう，コマンド出力に加えて，これらを付与して記録する．

このようなコマンド遠隔実行機能に加えて，コマンド編集時に Tab キーを入力すると，ログイン中のサーバにおけるコマンド名やファイルパス名を補完入力することができる．

4.2 SSH Kernel の設計と実装

ここでは，SSH Kernel の設計と実装にあたり，検討を行った諸々の項目について述べる．

4.2.1 Jupyter Notebook 上での遠隔コマンド実行

オペレータが対象サーバにログインし，「コマンド入力，実行，結果確認」のイテレーションステップを繰り返し，ログアウトする，という操作を前提とした「従来型手順書」のコマンド操作を「実行可能な手順書」でも同様に記述可能にするため，ログイン・ログアウトのための独自の命令 (%login, %logout) を導入した．

SSH Kernel ではログイン時に開設する SSH セッションを，ログアウトするまでのコマンド実行を通して使い回す．そのため，遠隔コマンド実行の度に SSH セッションを張り直すためにレスポンスが悪化するという IPython Kernel の問題は生じない．

4.2.2 ライブラリ選択

遠隔実行のための SSH クライアントは，できるだけ多くの OS で動作するよう Python による SSH クライアント実装 Paramiko [*3] を選択した．なお，遠隔実行中の例外処理や割り込みの実装を再利用するため，Paramiko を単に使うのではなく，Paramiko をベースとしたリモートコマンド実行機能をもつライブラリ Plumbum [*4] を併用した．

カーネルの実装にあたり，独自の命令の実装手段やカーネルが備えるべき共通機能を提供する，Metakernel [*5] を継承した．

4.2.3 実行コンテクストの引き継ぎ

カレントディレクトリ・環境変数という実行コンテクストを，コマンド実行をまたいで引き継ぐには 2 つの方法がある．

a) 遠隔サーバに状態を持つ方法である．これは，SSH セッション開設時に対象サーバのリモートシェルを起動しつづけることで状態を保持し，標準入出力経由でコマンド実行・結果取得を繰り返す方法である．

b) クライアントに状態を持つ方法である．状態を保持するリモートシェルは起動せず，代わりにコマンド実行前に実行コンテクストを復元し，実行後に採取する方法である．

a は，クライアント側の実装は一見シンプルであるが，コマンド実行が完了したことを標準出力の文字列マッチで判断するものである．これは誤マッチや文字化けに対する脆弱性を内包することになる．一方 b は，クライアント側の実装は比較的煩雑であるものの，文字列マッチの問題を生じない点で堅牢である．

SSH Kernel では b の方法を採り，実行コンテクストをクライアント側の SSH Kernel プロセスで管理する．

実行コンテクストの採取は，次のように実現する．

1. 遠隔実行するコマンドの末尾に環境変数を表示するコマンドを連結し，実行する
2. 遠隔でのコマンド実行終了時点の環境変数を，文字列パターンマッチにより採取し，記憶する．環境変数にはカレントディレクトリも含まれる

実行コンテクストの復元は，遠隔実行するコマンドの直前で，カレントディレクトリの移動および記憶した環境変数の宣言を行うことで実現する．

4.3 実装

SSH Kernel のソースコード規模と，動作確認環境を次に示す．

ソースコード規模

- SSH Kernel: 約 600 行
- テスト: 700 行

動作確認環境

- Python 3.6, 3.7
- IPython 7.1
- Ubuntu 16.04, 18.04

[*3] https://github.com/paramiko/paramiko/

[*4] https://github.com/tomerfiliba/plumbum/

[*5] http://calysto.github.io/metakernel/

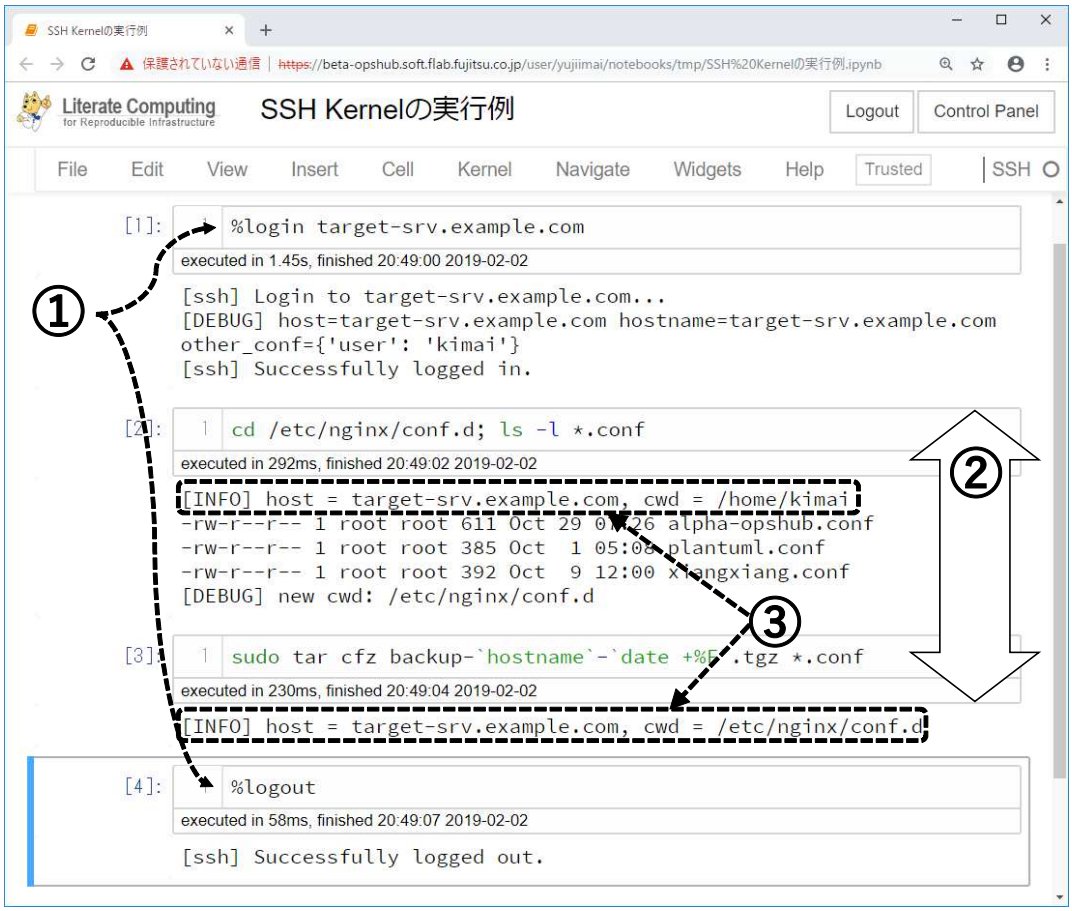

図 7　SSH Kernel と Jupyter Notebook を用いた「実行可能な手順書」の記述・実行例

5. 評価

SSH Kernel が課題を解消しているかを確かめるため，既存ツールとの機能比較およびトライアルユーザに対するアンケートを行った．

5.1 機能比較

図 8 は SSH Kernel と既存ツールの機能比較を行ったものである．

- 節 2.1 で述べた，人手を介さず実行でき文章として読みやすいという「実行可能な手順書」の特長を引き継ぐ
- 「配置 A」で生じていた，対象サーバへのソフトウェア導入の課題を解消する
- 「配置 B」で生じていた，遠隔コマンド実行に関する既存カーネルの課題を解消し，「従来型手順書」記載のコマンドがそのまま記述可能である

このように既存ツールの課題を解消する一方で，（vi, sudo などの）対話型コマンドがそのまま利用できず書き換えを要する点は，Jupyter Notebook を用いた「実行可能な手順書」に共通する課題である．

表 1　職種別の回答者数

職種	回答者数
システムエンジニア	2
ソフトウェアエンジニア	1
研究開発	4
合計	7

5.2 アンケート

SSH Kernel の有効性を確かめるため，SSH Kernel トライアルユーザへのアンケートを行った．

5.2.1 回答者属性

職種別の回答者数を 表 1 に示す．なお，職種や試用期間（最短 2 週間，最長 2 ヶ月）による回答の偏りは見られなかった．

5.2.2 既存カーネルとの比較

Jupyter Notebook と組み合わせて用いる，既存カーネルと SSH Kernel について，遠隔コマンド入力の容易さを尋ねた．

設問

SSH 端末 (Tera Term など) によるコマンド入力の容易さを 10 とした場合，次の各ツールの相対的な遠隔コマンド入力の容易さを入力して下さい

要件	Jupyter Notebook ＋SSH Kernel	Jupyter Notebook＋既存カーネル (IPython Kernel)	インフラ自動化ツール (Ansible・Chef)	「従来型手順書」＋SSH端末
機械的に実行可能	✓	✓	✓	不可
文章として読みやすい	✓	!ssh … への書き換えが冗長で読みにくい	文章としての可読性は低い	✓
シェルスクリプト以外のプログラミングスキルが不要	✓	Pythonスキル必要	Python, Rubyスキル必要	✓
対象サーバへのソフトウェア導入	SSHデーモン	SSHデーモン	・SSHデーモン ・Pythonインタプリタ (Ansible) ・Chefクライアントバイナリ (Chef)	SSHデーモン
「従来型手順書」に記述された遠隔コマンドを，そのまま記述可能	✓	書き換え必要	書き換え必要	✓
遠隔コマンド実行をまたぐ，実行コンテクストの引き継ぎ	✓	オペレータが明に与える	オペレータが明に与える	✓
対話型コマンド (vi, sudoなど) 利用可	不可	不可	不可	✓
SSH接続時，公開鍵認証に対応	✓	✓	✓	✓
SSH接続時，パスワード認証に対応	未対応	未対応	✓	✓

図 8　機能比較

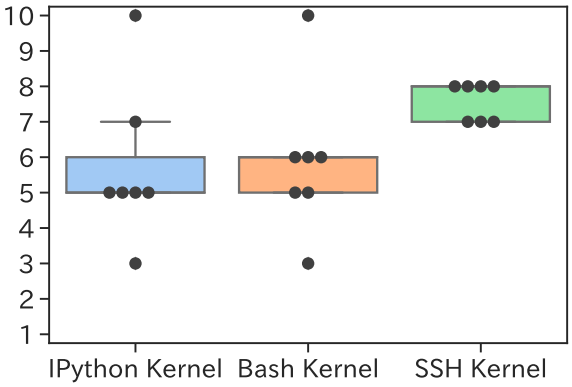

図 9　SSH 端末 (Tera Term など) によるコマンド入力の容易さを 10 とした場合，各ツールの相対的な遠隔コマンド入力の容易さ

結果を図 9 に示す．既存カーネルの相対的容易さが 5〜6 であるのに比べ，SSH Kernel は 7〜8 であるとの回答が得られた．ただし，SSH 端末と同程度の使い勝手 (縦軸の値が 10) には至っていない．

5.2.3　SSH 端末との比較

SSH Kernel が SSH 端末と同様の使い勝手を提供するかどうかを 3 段階で尋ねた．結果は表 2に示すように，「しばしば不都合を感じる」が大半を占めた．

続く設問では，不都合を感じる理由を尋ねた．

> 設問
>
> SSH 端末と比べて「不都合を感じる」場合，理由を教えて下さい（自由記述）

得られた回答のうち主なものを次に示す（「」は原文，〔〕は筆者注）

表 2　SSH Kernel は遠隔コマンドを，対話型シェルのように入力・実行できると感じますか（3 段階）

選択肢	回答数
1. 頻繁に不都合を感じる	1
2. しばしば不都合を感じる	5
3. 同程度	1
4. 回答なし	1

- 標準入力への入力を求める対話型コマンドに非対応である
  - 「対話型コマンド vi, more などが使えない」
  - 「対話（相手からの返答を受けて，臨機応変に対応するような”対話”）〔的なインタフェース〕にはなっていないと思う」
- セキュリティポリシーを変える必要がある
  - 「〔運用〕操作の多くは root 権限が必要だが，そのために遠隔マシンのセキュリティ設定を変えないといけない．下記いずれかが必要だが，Ubuntu の場合デフォルトではいずれも禁止されている．1) パスワード入力無しで sudo による昇格を実行できるようにする．2) root で遠隔ログインできるようにする」
- その他，対話型シェルとのギャップがある
  - 「〔Bash の〕Ctrl+p のヒストリ機能などが使えない」

5.3　本節のまとめ

機能比較では，2.1 節で述べた「実行可能な手順書」による利点を引き継ぎ，3 節で述べた課題を解消することを確かめた．

アンケートでは，既存カーネルに比べ遠隔コマンド入力

は容易であるという評価を得た．一方で，対話型コマンドが使えないこと，セキュリティポリシーを変える必要があることに不都合を感じるとの評価があった．

6. おわりに

本稿では，Jupyter Notebook を用いた「実行可能な手順書」を遠隔サーバ運用に適用する際の課題を解消するための Jupyter 拡張「SSH Kernel」を提案した．SSH Kernel は，SSH 端末による対話的操作の置き換えには未だ課題や制約が残るものの，既存のカーネルと比べ遠隔サーバ運用への「実行可能な手順書」の適用に有効であることを確かめた．開発した SSH Kernel はオープンソースとして公開した（ソースコード：`https://github.com/NII-cloud-operation/sshkernel`，Python パッケージ: `https://pypi.org/project/sshkernel`）．

今後は，sudo プロンプトへの対応，ログイン中のサーバごとに表示を変えることで対象サーバの取り違えを抑止する機能，さらにログイン中か否かをノートブック上でユーザに明に示す SSH 接続状態の可視化機能などを追加する予定である．

参考文献

[1] 政谷好伸，谷沢智史，横山重俊: インフラ・コード化の実践における IPython Notebook の適用 (サービスコンピューティング)，電子情報通信学会技術研究報告 = IEICE technical report : 信学技報， Vol. 115, No. 72, pp. 27–32 (2015).
[2] Masatani, Y.: Collaboration and automated operation as literate computing for reproducible infrastructure, *JupyterCon* (2017).
[3] NII Cloud Operation Team: Literate Computing for Reproducible Infrastructure, , available from ⟨https://literate-computing.github.io/⟩ (accessed 2019-02-01).
[4] GitLab: Runbooks, , available from ⟨https://docs.gitlab.com/ee/user/project/clusters/runbooks/⟩ (accessed 2019-02-01).
[5] Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B. E., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J. B., Grout, J., Corlay, S. et al.: Jupyter Notebooks-a publishing format for reproducible computational workflows., *ELPUB*, pp. 87–90 (2016).