

GitHubおよびJenkinsによる Sakaiシステム構築

2017年 3月21日

情報メディア教育研究センター

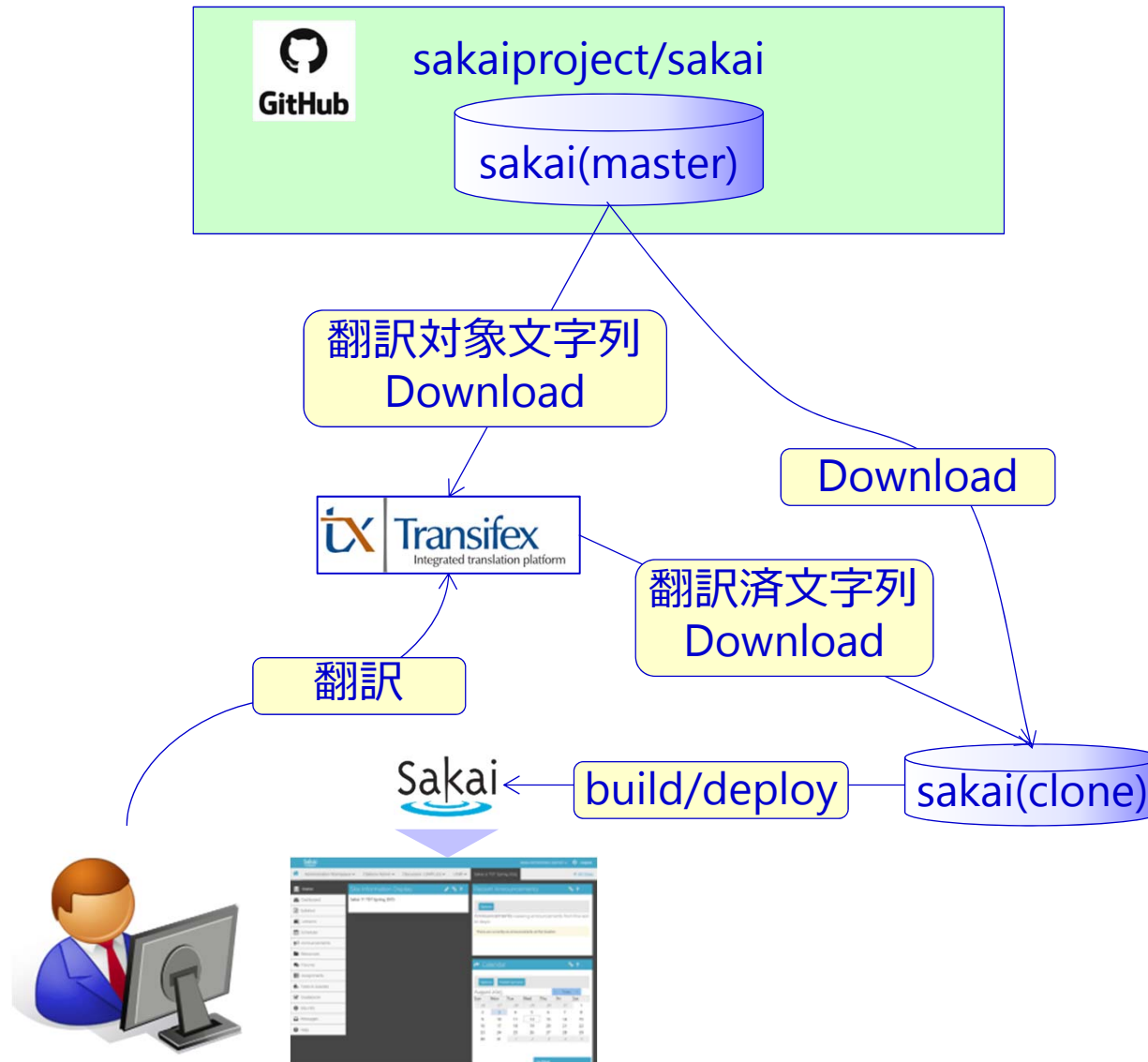
常盤 祐司

yuji.tokiwa.dc@hosei.ac.jp

Motivation

日本語に翻訳した文字列やメッセージが、
実際のWebアプリケーションで、
どのように表示されるかを即時で確認したい。

Sakai 翻訳概要



Git/GitHubおよびJenkinsの必要性

■ Git/GitHub

- SakaiではマスターソースをGitHubで管理するようになり、ソースのダウンロードおよびパッチの適用(Pull Request)にGitとGitHubが必要となる。

■ Jenkins

- Sakaiではビルドだけでも数分以上かかり、一連のプロセスをマニュアルで実行するには時間がかかり過ぎる。
- 複数のプロセスを逐次実行する場合、
 - マニュアルでは、手順を誤ることがある。
 - Shell Scriptでは、あるプロセスの終了後に次のプロセスを起動する制御が難しいことがある。

Git (ギット)

Awareness	バージョン管理システム 2005年にLinus Torvaldsが1,244行書き、 同年7月から濱野純氏が引き継ぎ、現在に至る
Information	Command Line Interface
Personal	OSSコミュニティでは標準になりつつある
Management	ネットや各種参考書で利用法をマスタできる
Consequence	使えないとコミュニティへのContributionができない
Collaboration	各国の開発者が利用している
Refocusing	無償のGitHubを活用し試用する



CBAM	
•Awareness (気づき)	それは何か？
•Information (機能)	どのように動くか？
•Personal (意義)	自分にとって役立つか？
•Management (操作)	どうすれば利用法をマスタできるか？
•Consequence (重要性)	価値があるか？
•Collaboration (協調)	他の人はどうしているか？
•Refocusing (再確認)	うまく使うために他にすることはないか？

GitHub (ギットハブ)

Awareness	クラウド上のGitリポジトリ、GitHub社が運営しているアカデミックは申請すれば無償で利用できる
Information	Web based UI, Gitと連携する
Personal	OSSコミュニティでは標準になりつつある
Management	ネットや各種参考書で利用法をマスタできる
Consequence	使えないとコミュニティへのContributionができない
Collaboration	各国の開発者が利用している
Refocusing	無償のGitHubを活用し試用する



CBAM

•Awareness (気づき)	それは何か？
•Information (機能)	どのように動くか？
•Personal (意義)	自分にとって役立つか？
•Management (操作)	どうすれば利用法をマスタできるか？
•Consequence (重要性)	価値があるか？
•Collaboration (協調)	他の人はどうしているか？
•Refocusing (再確認)	うまく使うために他にすることはないか？

Jenkins (ジェンキンス)

Awareness Continuous Integration Tool
システム構築手順をプログラムして実行できる
川口耕介氏を中心として開発されている



Information Web based UI, プラグインで機能を拡張する

Personal OSSコミュニティでは標準になりつつある



Management ネットや各種参考書で利用法をマスタできる

Consequence 使わないとシステム構築作業に時間がかかる

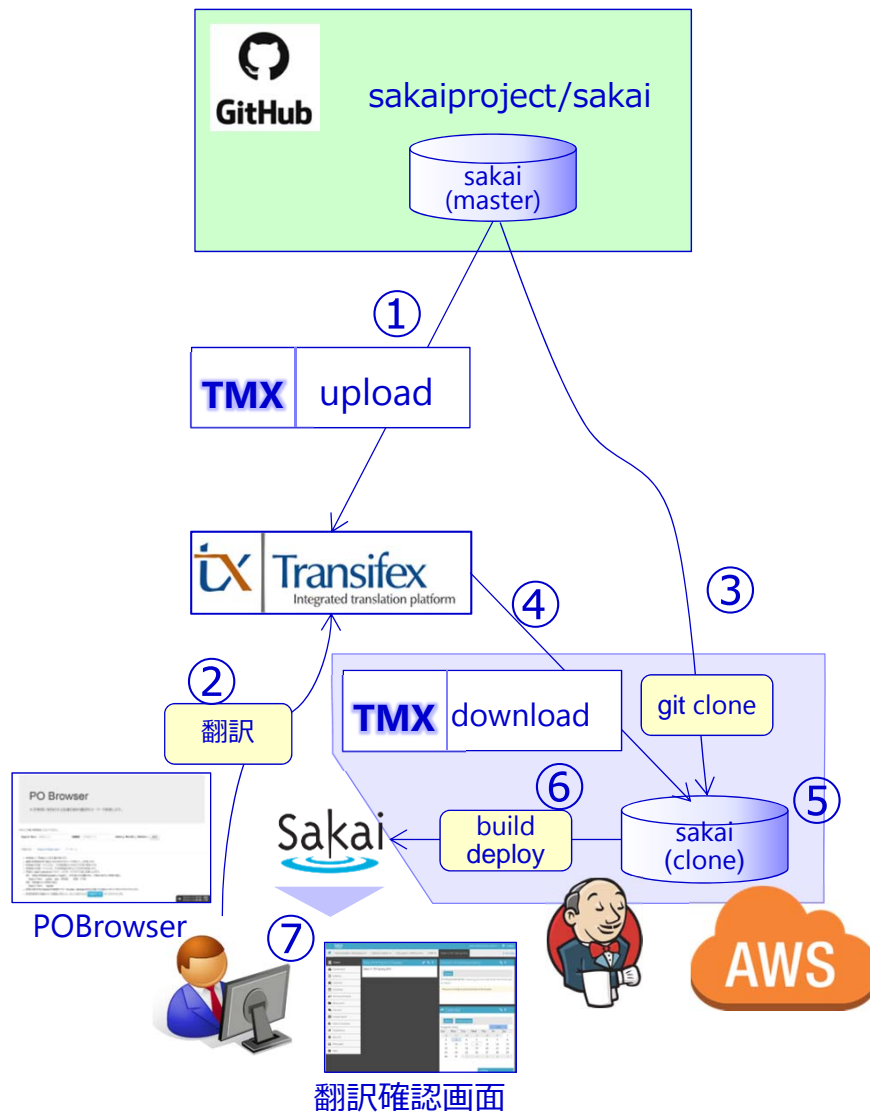
Collaboration 各国の開発者が利用している

Refocusing 無償のJenkinsを活用し試用する

CBAM

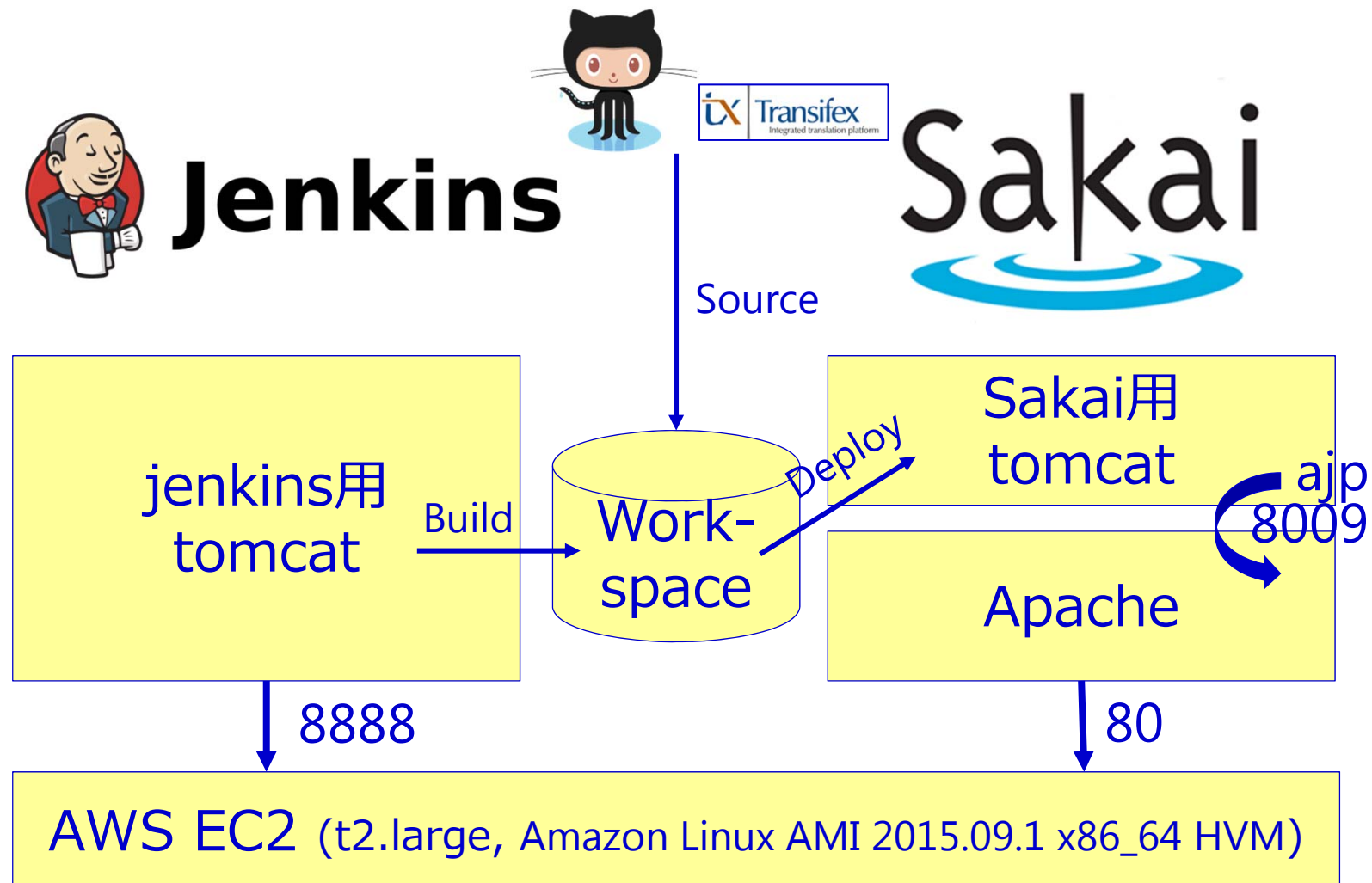
- | | |
|---------------------|---------------------|
| •Awareness (気づき) | それは何か？ |
| •Information (機能) | どのように動くか？ |
| •Personal (意義) | 自分にとって役立つか？ |
| •Management (操作) | どうすれば利用法をマスタできるか？ |
| •Consequence (重要性) | 価値があるか？ |
| •Collaboration (協調) | 他の人はどうしているか？ |
| •Refocusing (再確認) | うまく使うために他にすることはないか？ |

Ja Sakaiにて構築したSakai 翻訳環境

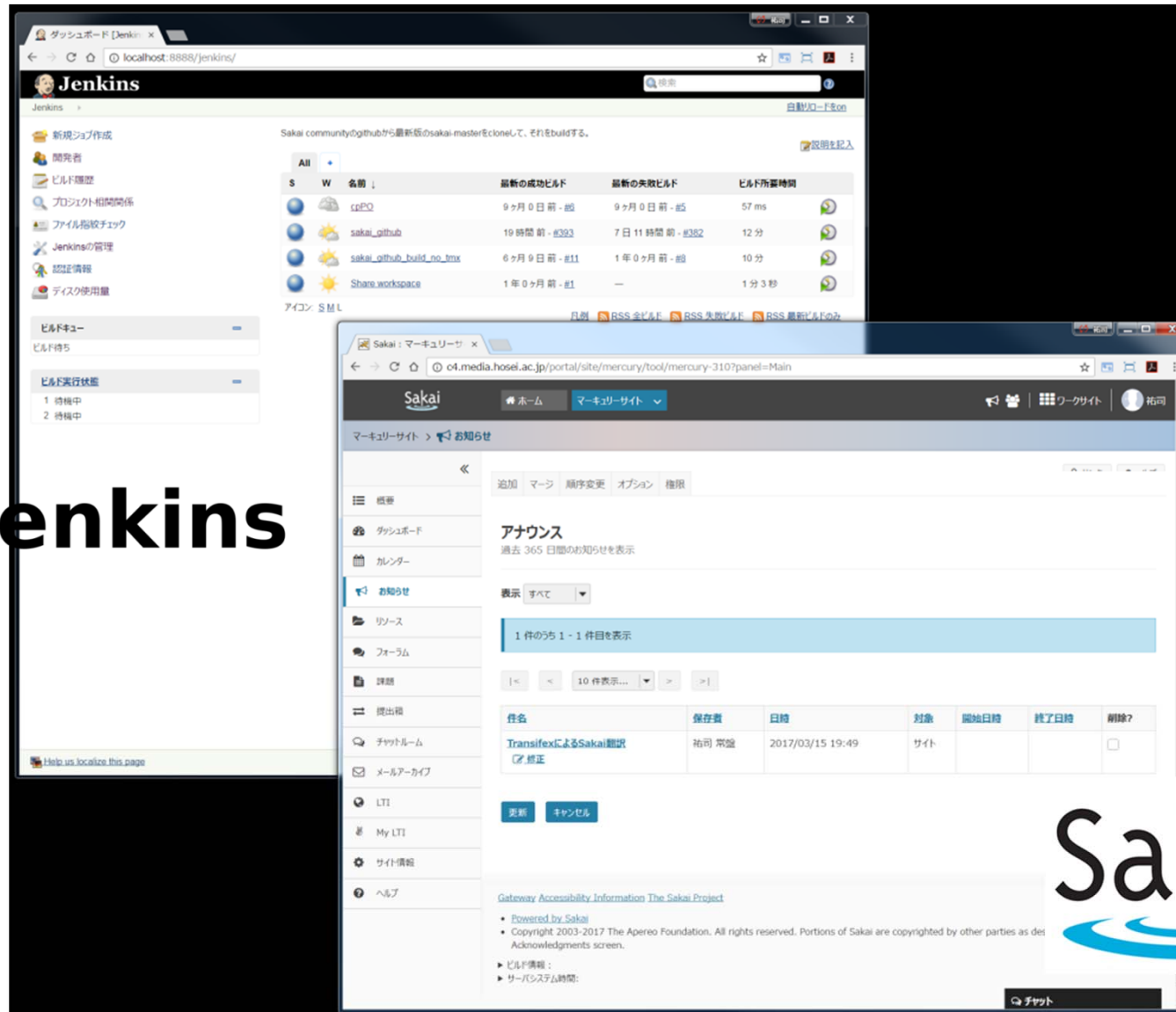


	翻訳手順	ツール	備考
1	GitHubからSakai Masterソースコードの翻訳対象箇所のみをTransifexにUpload	JaSakai/l10n	
2	用語集検索ツールのPOBrowserを参照しTransifexにて翻訳	Transifex, POBrowser	
3	GitHubからSakai Masterソースコードを検証用サーバにDownload	Git	Jenkins
4	TransifexからPOファイルを検証用サーバにDownload	JaSakai/l10n	Jenkins
5	POファイルをソースコードに展開	JaSakai/l10n	Jenkins
6	Build - Deploy - Tomcat起動	Shellscript	Jenkins
7	Web BrowserにてBuildしたアプリケーションを確認		

検証用サーバシステム構成



翻訳環境事例



The image shows two overlapping browser windows. The background window is the Jenkins dashboard, displaying a table of build jobs. The foreground window is the Sakai CMS interface, showing a notification for a translation task.

S	W	名前	最新の成功ビルド	最新の失敗ビルド	ビルド所要時間
		gitPO	9ヶ月0日前 - #5	9ヶ月0日前 - #5	57 ms
		sakai_github	19時間前 - #393	7日11時間前 - #392	12分
		sakai_github_build_no_tm	6ヶ月9日前 - #11	1年0ヶ月前 - #8	10分
		Share_workspace	1年0ヶ月前 - #1	-	1分3秒

件名	保存者	日時	対象	開始日時	終了日時	削除?
TransifexによるSakai翻訳 修正	祐司 常盤	2017/03/15 19:49	サイト			<input type="checkbox"/>



Jenkins

Sakai

オンデマンドオペレーション

Jenkins Webコンソールから即時でビルド実行を指示できる



Sakai communityのgithubから最新版のsakai-masterをcloneして、それをbuildする。

S	W	名前 ↓	最新の成功ビルド	最新の失敗ビルド	ビルド所要時間
		cpPQ	8ヶ月 28日 前 - #6	8ヶ月 28日 前 - #5	57 ms
		sakai_github	54分 前 - #391	6日 6時間 前 - #382	11分
			6ヶ月 8日 前 - #11	1年 0ヶ月 前 - #8	10分
			1年 0ヶ月 前 - #1	—	1分 3秒

アイコン: S M L

ビルドキュー
ビルド待ち

ビルド実行状態
1 待機中
2 待機中



Jenkins sakai_github #392

コンソール出力

Started by user anonymous
 Building in workspace /root/.jenkins/sharedspace/sharedWS
 > git rev-parse --is-inside-work-tree # timeout=10
 Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/sakaiproject/sakai.git # timeout=10
 Fetching upstream changes from https://github.com/sakaiproject/sakai.git
 > git --version # timeout=10
 > git -c core.askpass=true fetch --tags --progress https://github.com/sakaiproject/sakai.git +refs/heads/*:refs/remotes/origin/*
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
 > git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
 Checking out Revision 4c8573c460e032d4f3f7bc31b5480ac048c0a87b (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 4c8573c460e032d4f3f7bc31b5480ac048c0a87b
 > git rev-list 4c8573c460e032d4f3f7bc31b5480ac048c0a87b # timeout=10
 [sharedWS] \$ /bin/sh -xe /opt/tomcat_jenkins/temp/hudson7048460734330129477.sh
 + /usr/bin/python /root/.jenkins/sharedspace/sharedWS/110n/tmx.py download -u -l ja
 Pulling translations for resource sakai-trunk.polls (source: templates/polls.pot)
 -> ja: ja/polls.po
 Pulling translations for resource sakai-trunk.msgcntr (source: templates/msgcntr.pot)
 -> ja: ja/msgcntr.po
 Pulling translations for resource sakai-trunk.site (source: templates/site.pot)
 -> ja: ja/site.po
 Pulling translations for resource sakai-trunk.syllabus (source: templates/syllabus.pot)
 -> ja: ja/syllabus.po
 Pulling translations for resource sakai-trunk.user (source: templates/user.pot)
 -> ja: ja/user.po

Jenkins環境設定



The image shows the Jenkins web interface. On the left is a navigation menu with items like '新規ジョブ作成', '開発者', 'ビルド履歴', 'プロジェクト相関関係', 'ファイル指紋チェック', 'Jenkinsの管理', '認証情報', and 'ディスク使用量'. The main area is titled 'Jenkinsの管理' and contains a warning about version 2.50 and several icons for system settings, global security, and configuration refresh. An inset window shows the '設定' (Settings) page, with a blue arrow pointing to the 'システムの設定' (System Settings) section. Another blue arrow points from the 'インストール済みJDK...' button in the 'JDK' section of the settings page to a detailed view of the installed JDKs.

JDK

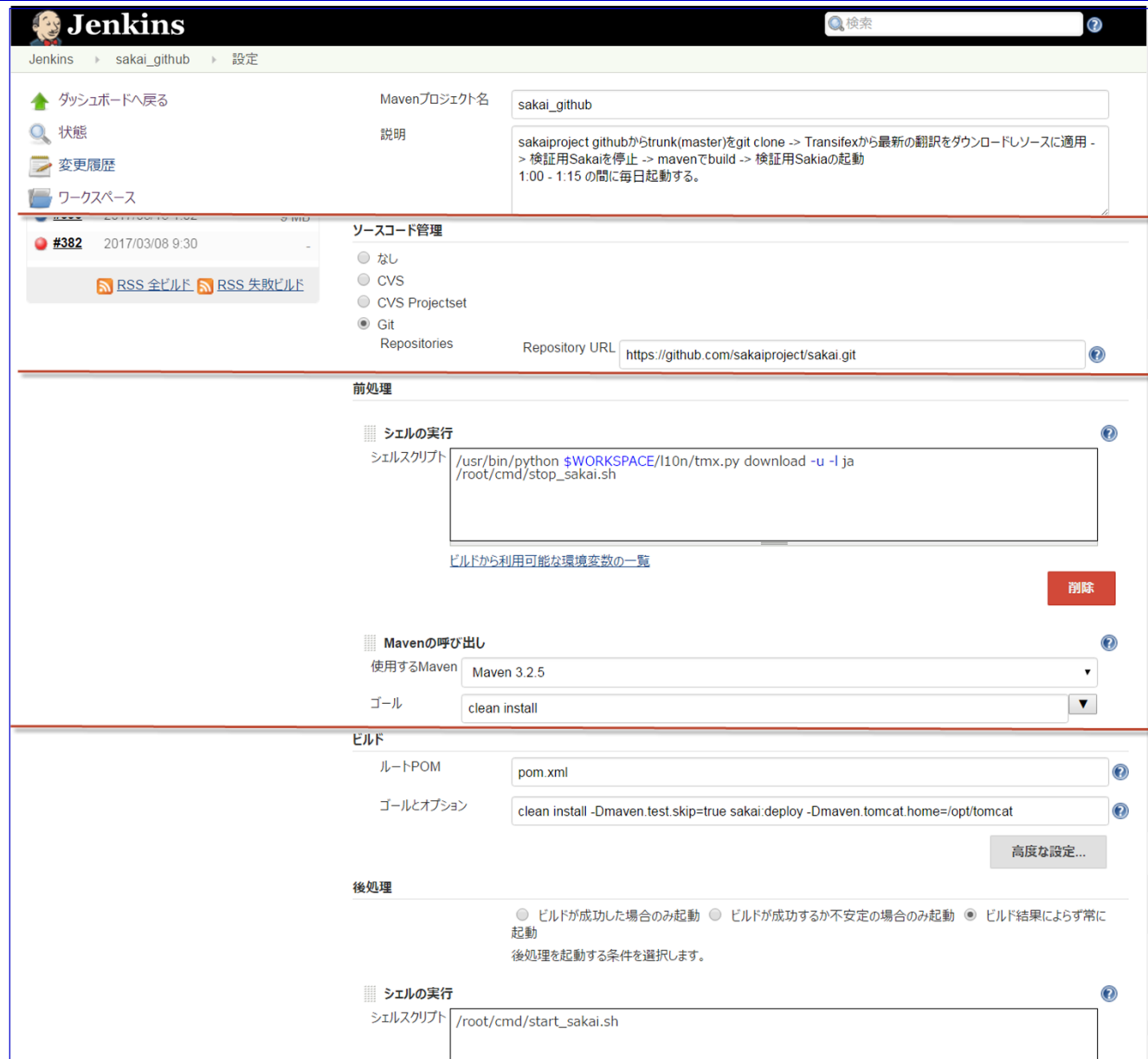
インストール済みJDK

JDK 名前	JAVA_HOME
jdk 1.8.0.45	/opt/java

Sakai ビルド手順

1. 最新のSakaiソースを `git clone https://github.com/sakaiproject/sakai.git` にて `{$WORKSPACE}` にクローンする。
2. Transifexからサーバに最新の翻訳ファイルをすべてダウンロードするために、`{$WORKSPACE}/l10n` ディレクトリにて `"python tmx.py download -u -l ja"` というコマンドを実行する。
3. SakaiのWebコンテナであるtomcatを停止するために、`"stop_sakai.sh"` というシェルスクリプトを実行する。
4. Sakaiを構成するモジュールのpom.xmlを生成するために、`{$WORKSPACE}/master` ディレクトリにて、`"mvn clean install"` というコマンドを実行する。
5. Sakaiを構成するモジュールをビルドするために、`{$WORKSPACE}` ディレクトリにて、`"mvn clean install -Dmaven.test.skip=true sakai:deploy -Dmaven.tomcat.home=/opt/tomcat"` というコマンドを実行する。
6. SakaiのWebコンテナであるtomcatを起動するために、`"start_sakai.sh"` というシェルスクリプトを実行する。

Jenkins ジョブ設定



The screenshot shows the Jenkins configuration page for a job named 'sakai_github'. The page is divided into several sections:

- ダッシュボードへ戻る**: Home button.
- 状態**: Job status.
- 変更履歴**: Change history.
- ワークスペース**: Workspace.
- Mavenプロジェクト名**: sakai_github
- 説明**: sakaiproject githubからtrunk(master)をgit clone -> Transifexから最新の翻訳をダウンロードしソースに適用 -> 検証用Sakaiを停止 -> mavenでbuild -> 検証用Sakiaの起動 1:00 - 1:15 の間に毎日起動する。
- ソースコード管理**:
 - Repository type: Git
 - Repository URL: https://github.com/sakaiproject/sakai.git
- 前処理**:
 - シェルの実行**:

```

/usr/bin/python $WORKSPACE/l10n/tmx.py download -u -l ja
/root/cmd/stop_sakai.sh

```
 - Mavenの呼び出し**:
 - 使用するMaven: Maven 3.2.5
 - ゴール: clean install
- ビルド**:
 - ルートPOM: pom.xml
 - ゴールとオプション: clean install -Dmaven.test.skip=true sakai:deploy -Dmaven.tomcat.home=/opt/tomcat
- 後処理**:
 - ビルドが成功した場合のみ起動
 - ビルドが成功するか不安定の場合のみ起動
 - ビルド結果によらず常に起動
- シェルの実行**:

```

/root/cmd/start_sakai.sh

```

中略

中略

中略

Plugin

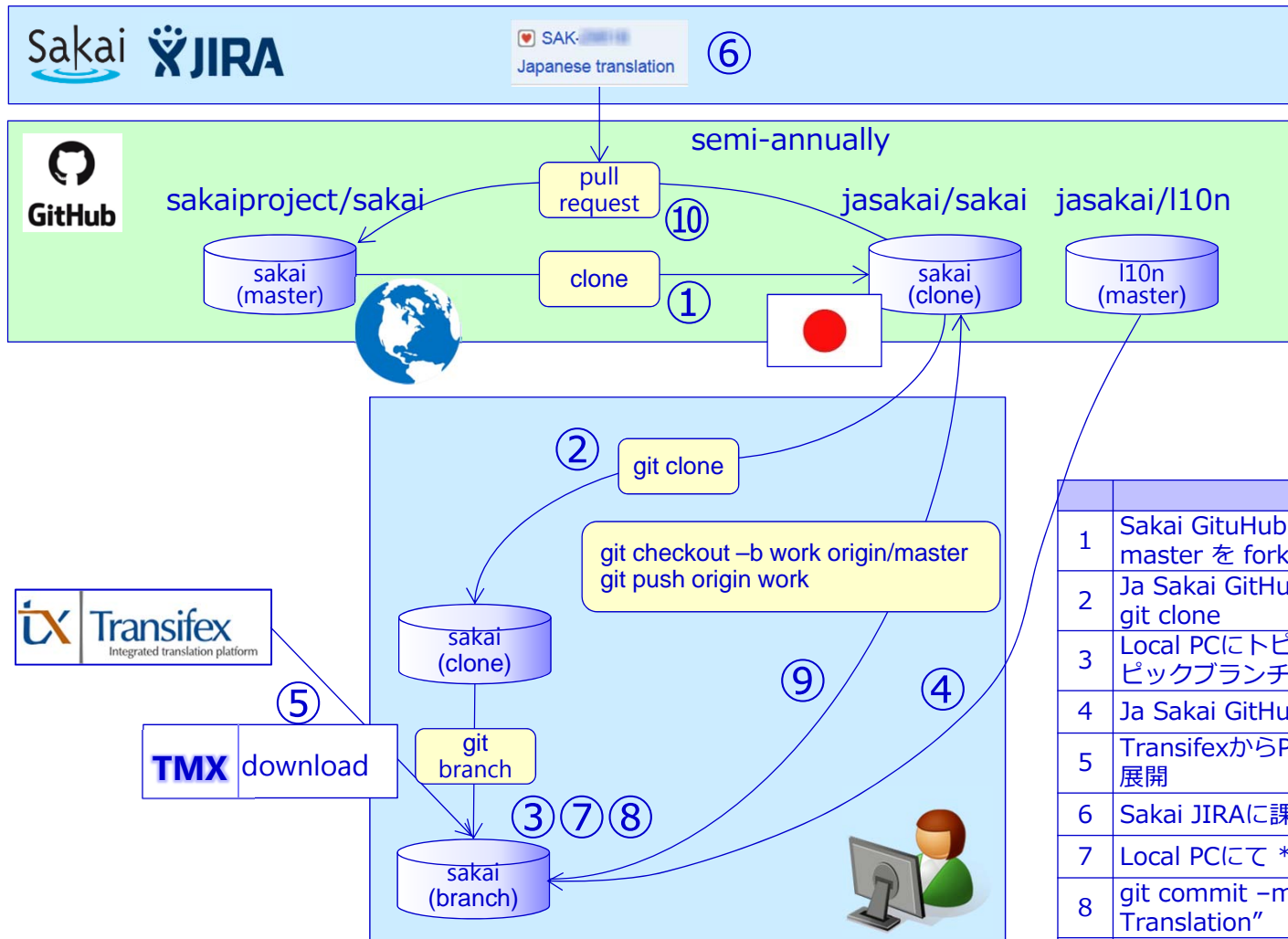
プラグイン名	概要
Discard Old Build	任意の世代までを残し、古いビルド履歴を削除する。
disk-usage	ディスク使用量をブラウザで確認する。
Git	Gitを利用する。
Maven	Mavenを利用する。
Shared Workspace	複数のジョブで Workspace を共有する。

TIPS

- JenkinsとSakaiは共にtomcatで稼働するが、Sakaiを停止して再起動するため、tomcatは共通とせず、それぞれのtomcatをSakai 8080, Jenkins 8888 にて起動している。
- PCからは、ssh接続時に8888をport forwarding し、<http://localhost:8888/jenkins> にてJenkinsのWebページを表示している。
- JenkinsからShell ScriptでSakaiのコンテナであるtomcatを起動すると、数分でtomcat が停止してしまう。Jenkins用のtomcatにて `tomcat_jenkins/bin/setenv.sh` を下記のように設定し、実行権限755を与えておく。

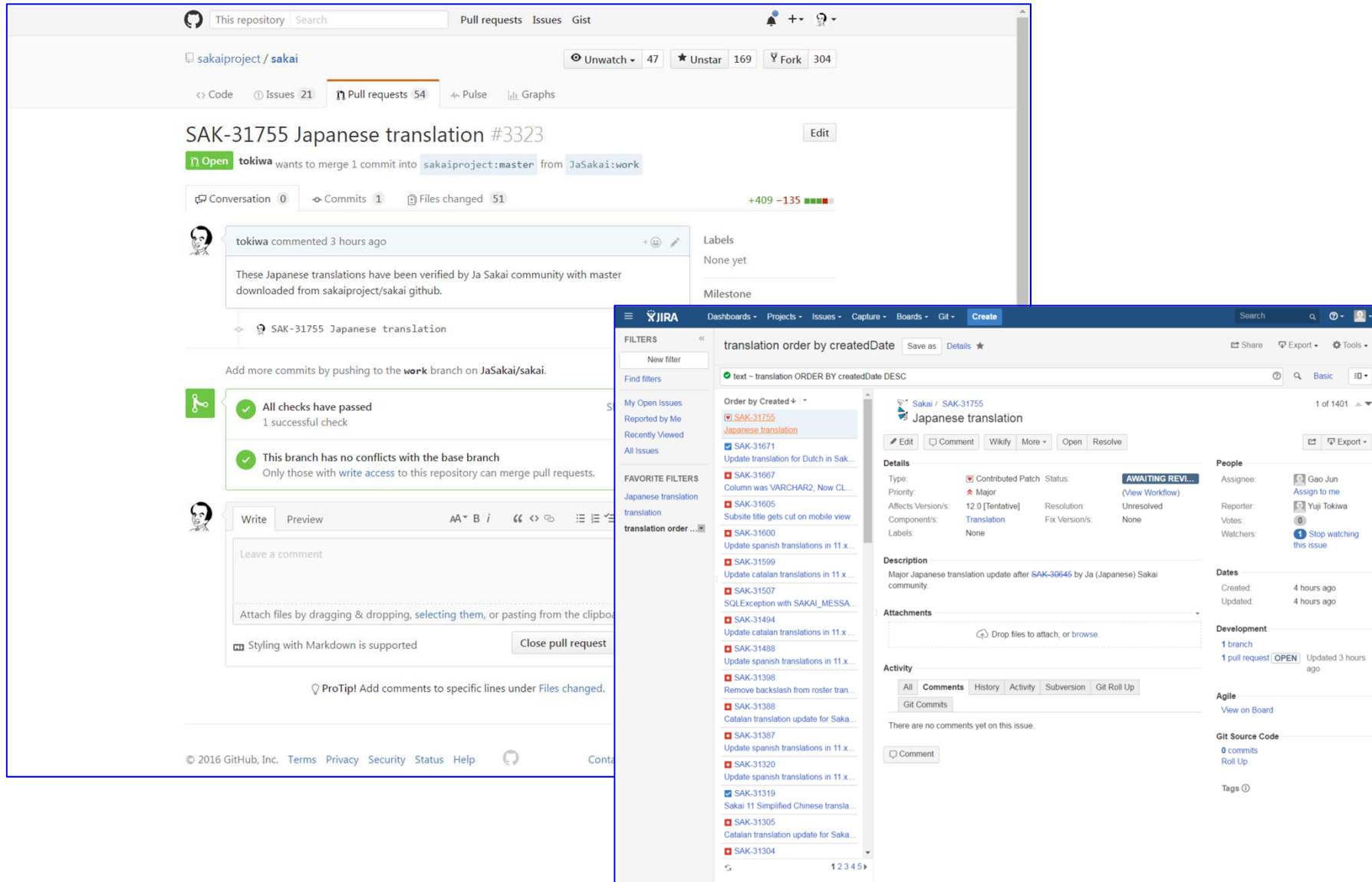
```
export JAVA_OPTS="-Dhudson.util.ProcessTree.disable=true"
```
- JenkinsはジョブごとにWorkspaceが与えられるが、翻訳の適用の有無などで同じWorkspaceを参照するためには、Shared Workspaceプラグインを追加すればよい。

Pull Request 概要



手順	
1	Sakai GituHubからJa Sakai GitHubに Sakai master を fork
2	Ja Sakai GitHubからLocal PC に Sakai master を git clone
3	Local PCにトピックブランチを作成し、HEADをトピックブランチに切替
4	Ja Sakai GitHubからLocal PC に l10n を git clone
5	TransifexからPOファイルをLocal PCにDownloadし展開
6	Sakai JIRAに課題を作成し、SAK-nnnnnを作成
7	Local PCにて *_ja.propertiesのみ git add
8	git commit -m "SAK-nnnnn Japanese Translation"
9	Ja Sakai GitHubに翻訳したソースコードをgit push
10	Ja Sakai GitHubにてSakai GitHubにpull request
	その後Sakai管理者がコミットしてソースコードに翻訳が反映される

Pull Request 事例



The image shows two overlapping screenshots. The background screenshot is a GitHub Pull Request for repository sakaiproject/sakai, titled "SAK-31755 Japanese translation #3323". It shows the pull request details, including the author tokiwa, the target branch sakaiproject:master, and the source branch JaSakai:work. The pull request is open and has 51 files changed. A comment from tokiwa states: "These Japanese translations have been verified by Ja Sakai community with master downloaded from sakaiproject/sakai github." The pull request status shows "All checks have passed" and "This branch has no conflicts with the base branch".

The foreground screenshot is a JIRA issue view for SAK-31755, titled "Japanese translation". The issue is in the "AWAITING REVL..." status. The description reads: "Major Japanese translation update after SAK-30645 by Ja (Japanese) Sakai community." The issue is assigned to Gao Jun and reported by Yuji Tokiwa. The issue has 1 pull request linked to it, which is in the "OPEN" status. The JIRA interface also shows a list of related issues and a sidebar with filters.

Pull Request 詳細手順書

1. sakaiproject/sakai github からJaSakai githubにforkする。右上にあるforkをクリックすると宛先リポジトリ選択画面が表示される。
→ JaSakai/sakaiが生成される。
 2. JaSakai/sakaiからPCにcloneする。 `git clone https://github.com/JaSakai/sakai.git`
 3. PCにてソースコードホームにcdする。
 4. `git branch -a`, `git status -s` などでリポジトリを確認する。
 5. トピックブランチworkを `git checkout -b work origin/master` にて作成する。
 6. ブランチを確認する。 `git branch -a`
 7. sakaiproject/l10nからl10nツールをダウンロードし、ソースコードホームにl10nというディレクトリで配置する。
`git clone https://github.com/JaSakai/l10n.git`
 8. l10nディレクトリにcdし `python tmx.py init` を実行する。→ .pyc、templates ディレクトリが生成される。
 9. templateディレクトリにpotファイルが生成されていることを確認する。
 10. `python tmx.py download -u -l ja` を実行する。→ ja ディレクトリが生成され、TransifexからPOファイルがダウンロードされる。
 11. ソースコードホームにcdする。
 12. `git status -s` で*_ja.propertiesファイルが変更、追加されていることを確認する。
 13. `git add ¥*_ja.properties` (¥はバックスラッシュ)で 日本語関連のpropertiesファイルをgit管理対象に含める。
 14. `git status -s` で確認する。
 15. Sakai JIRAを作成し、SAK-nnnnnの番号を得る。AssigneeにはGao Junを設定する。
 16. `git commit` する。 `git commit -m "SAK-nnnnn Japanese Transalation"`
 17. `git push origin work` でgithubのworkにl10nが適用されたコードをJa Sakai githubに追加する。
 18. JaSakai githubにアクセスし、workブランチに切り替え、New pull request ボタンをクリックする。
 19. Comparing Changes ページで*_ja.properties のみが対象になっていることを確認する。
- * pull request をするとgithubにてTravis CIが自動実行されるため20分程度statusはpendingとなる。

