法政大学
HOSEI University

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

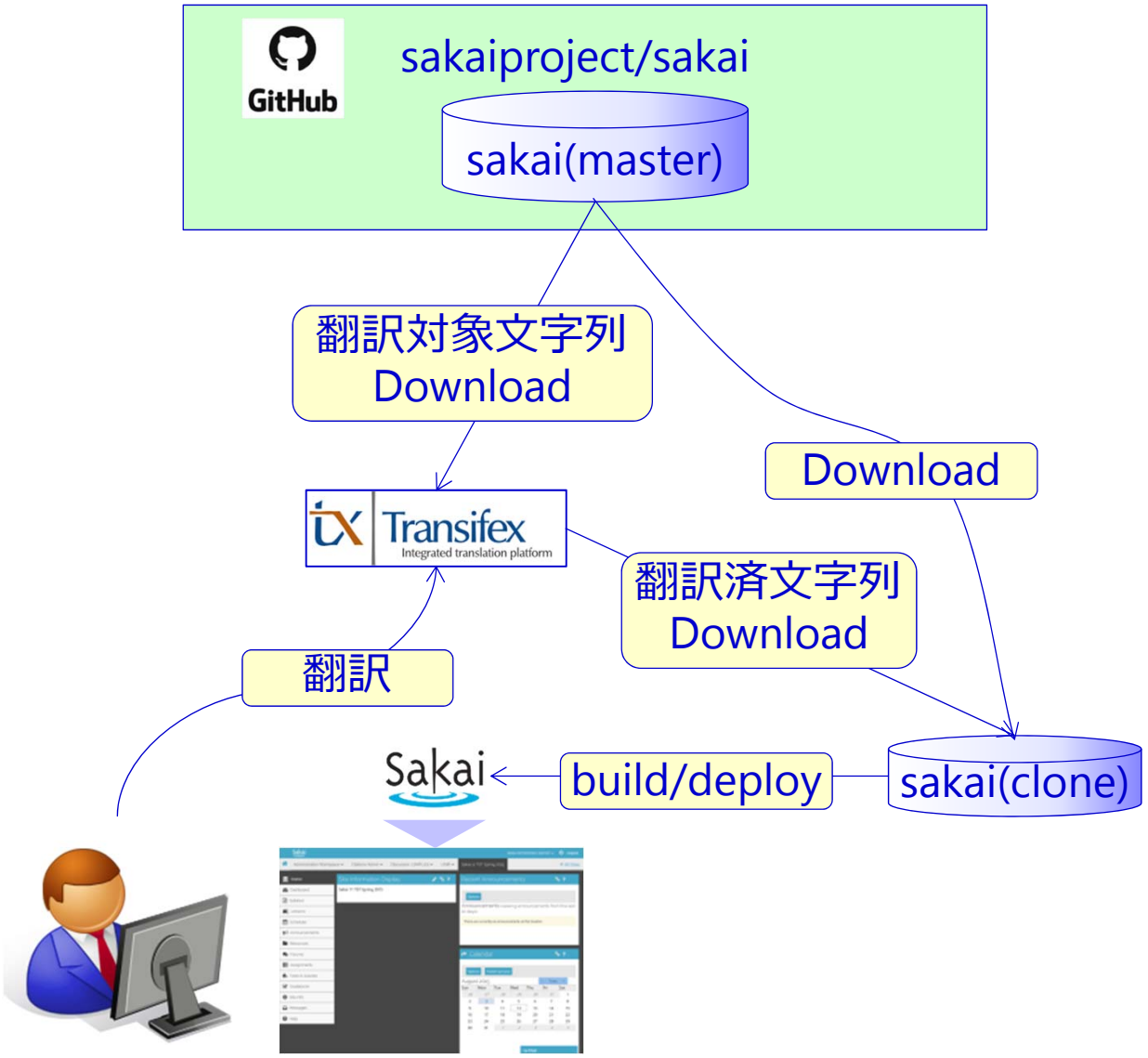
Motivation

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GutHub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）

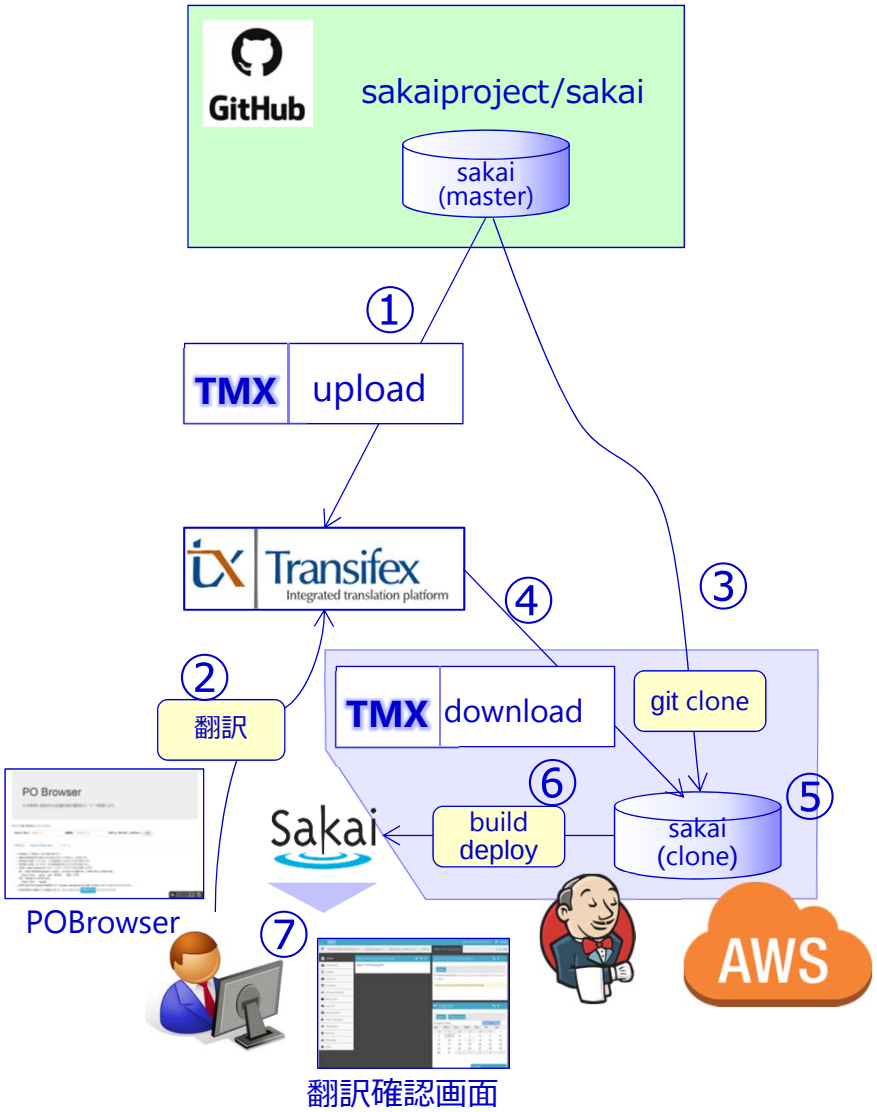
Awareness	Continuous Integration Tool システム構築手順をプログラムして実行できる 川口耕介氏を中心として開発されている
Information	Web based UI, プラグインで機能を拡張する
Personal	OSSコミュニティでは標準になりつつある
Management	ネットや各種参考書で利用法をマスタできる
Consequence	使わないとシステム構築作業に時間がかかる
Collaboration	各国の開発者が利用している
Refocusing	無償のJenkinsを活用し試用する

Jenkins

CBAM

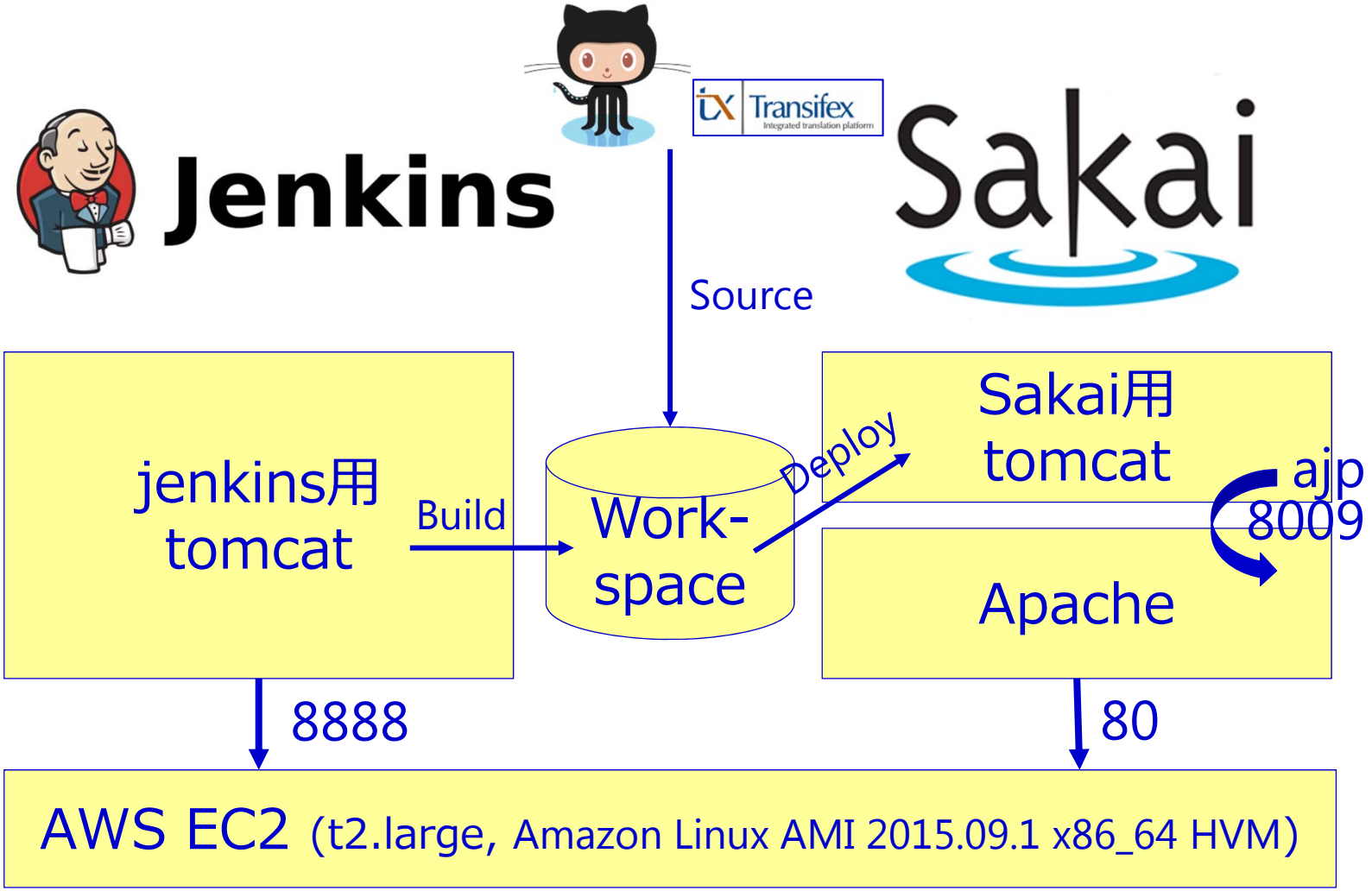
•Awareness (気づき)	それは何か？
•Information (機能)	どのように動くか？
•Personal (意義)	自分にとって役立つか？
•Management (操作)	どうすれば利用法をマスタできるか？
•Consequence (重要性)	価値があるか？
•Collaboration (協調)	他の人はどうしているか？
•Refocusing (再確認)	うまく使うために他にすることはないか？

Ja Sakaiにて構築したSakai 翻訳環境

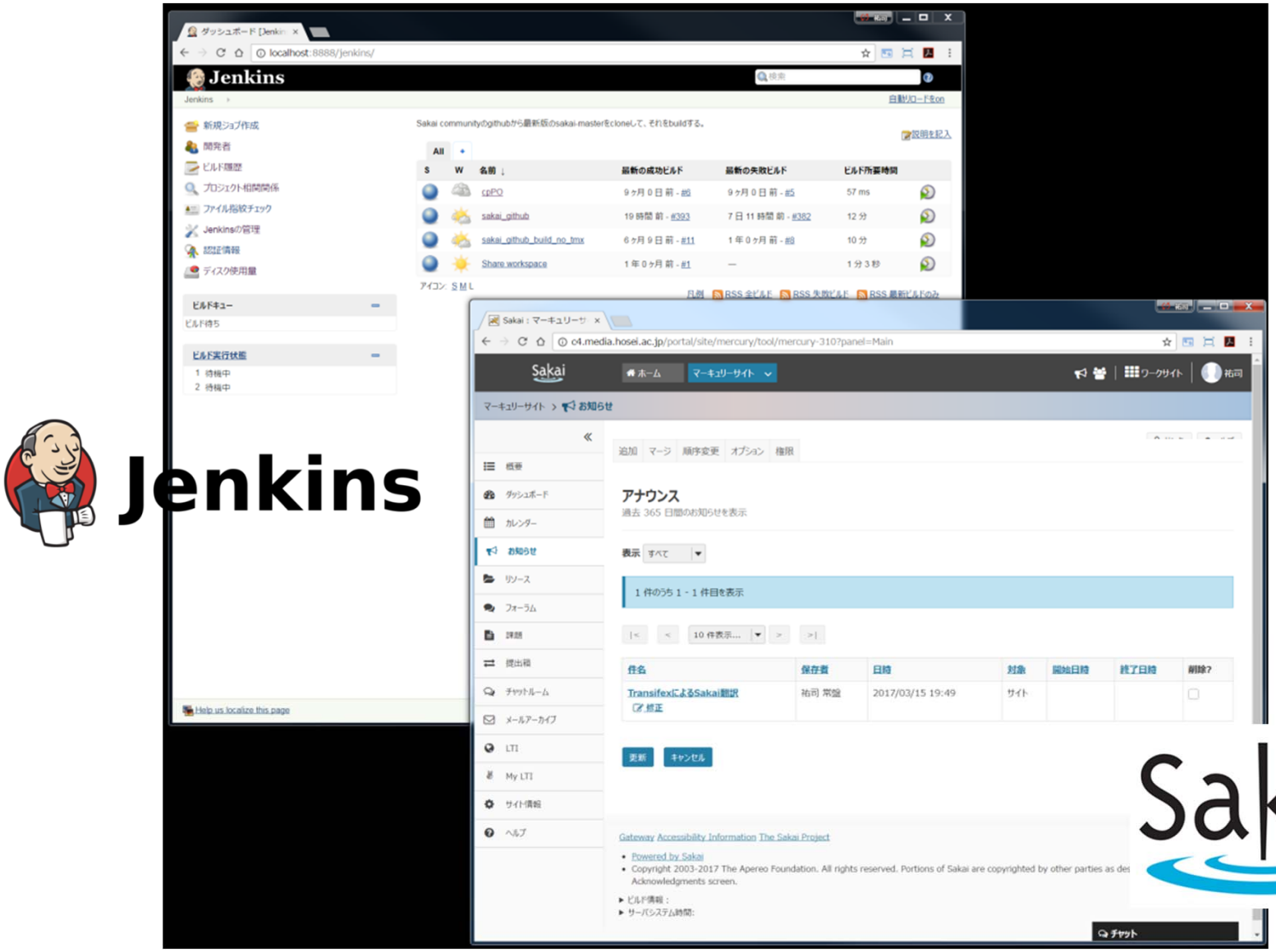


	翻訳手順	ツール	備考
1	GitHubからSakai Masterソースコードの翻訳対象個所のみをTransifexにUpload	JaSakai/ l10n	
2	用語集検索ツールのPOBrowserを参照しTransifexにて翻訳	Transifex, POBrowser	
3	GitHubからSakai Masterソースコードを検証用サーバにDownload	Git	Jenkins
4	TransifexからPOファイルを検証用サーバにDownload	JaSakai/ l10n	Jenkins
5	POファイルをソースコードに展開	JaSakai/ l10n	Jenkins
6	Build - 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



オンデマンドオペレーション

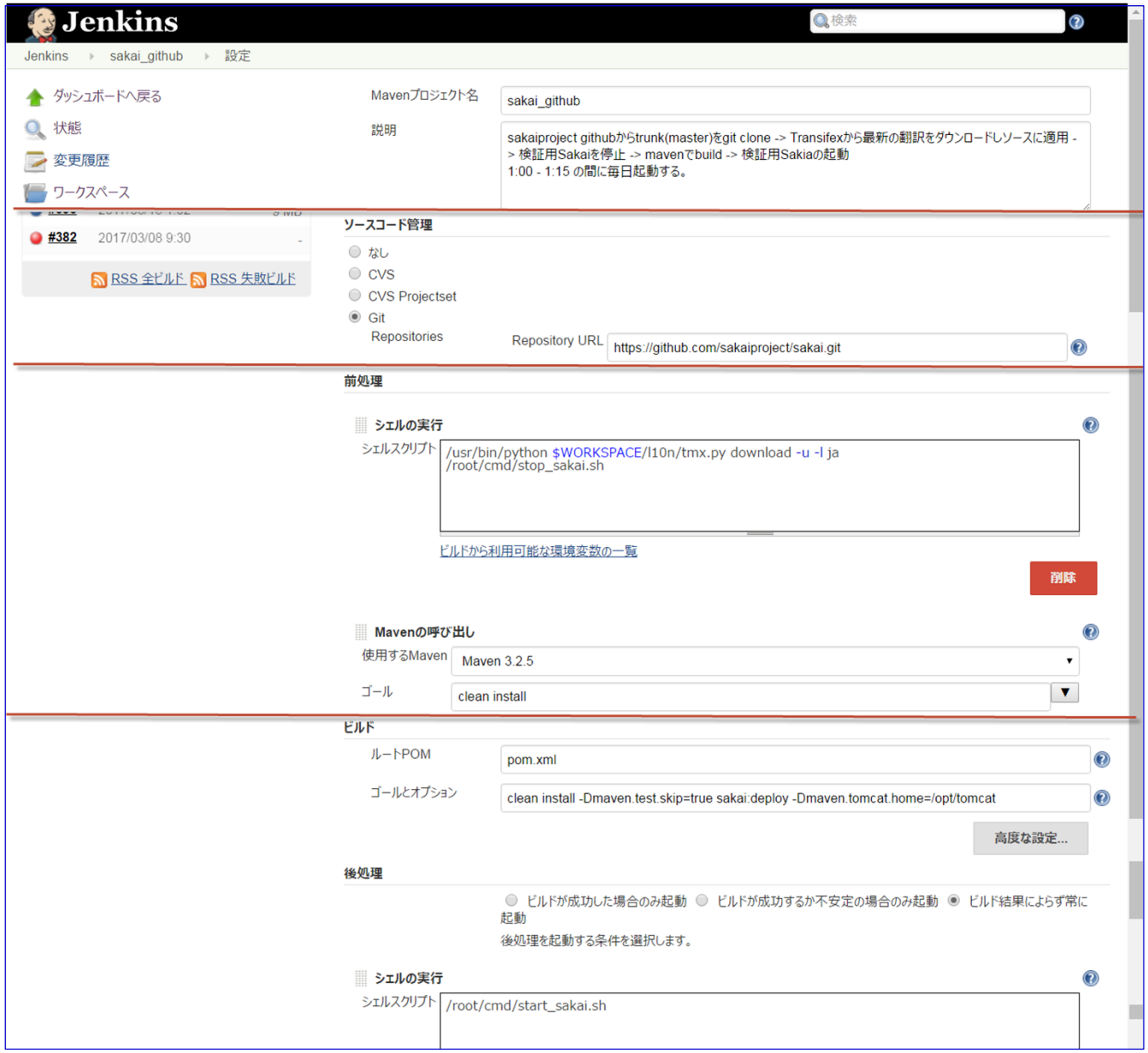
Jenkins Webコンソールから即時でビルド実行を指示できる

Jenkins環境設定

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



中略

中略

中略

Plugin

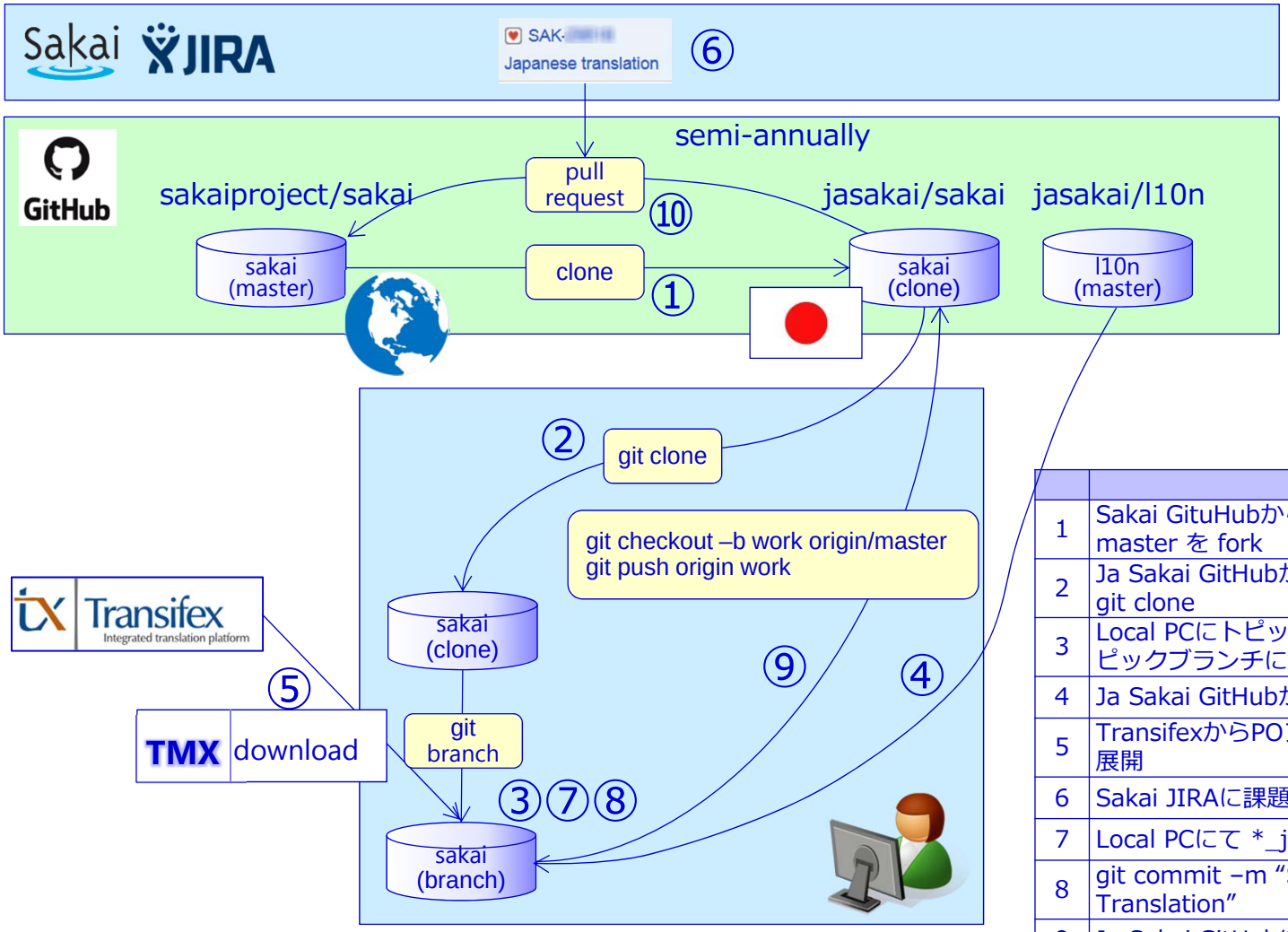
プラグイン名	概要
Discard Old Build	任意の世代までを残し、古いビルド履歴を削除する。
disk-usage	ディスク使用量をブラウザで確認する。
Git	Gitを利用する。
Maven	Mavenを利用する。
Shared Workspace	複数のジョブで Workspace を共有する。

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。

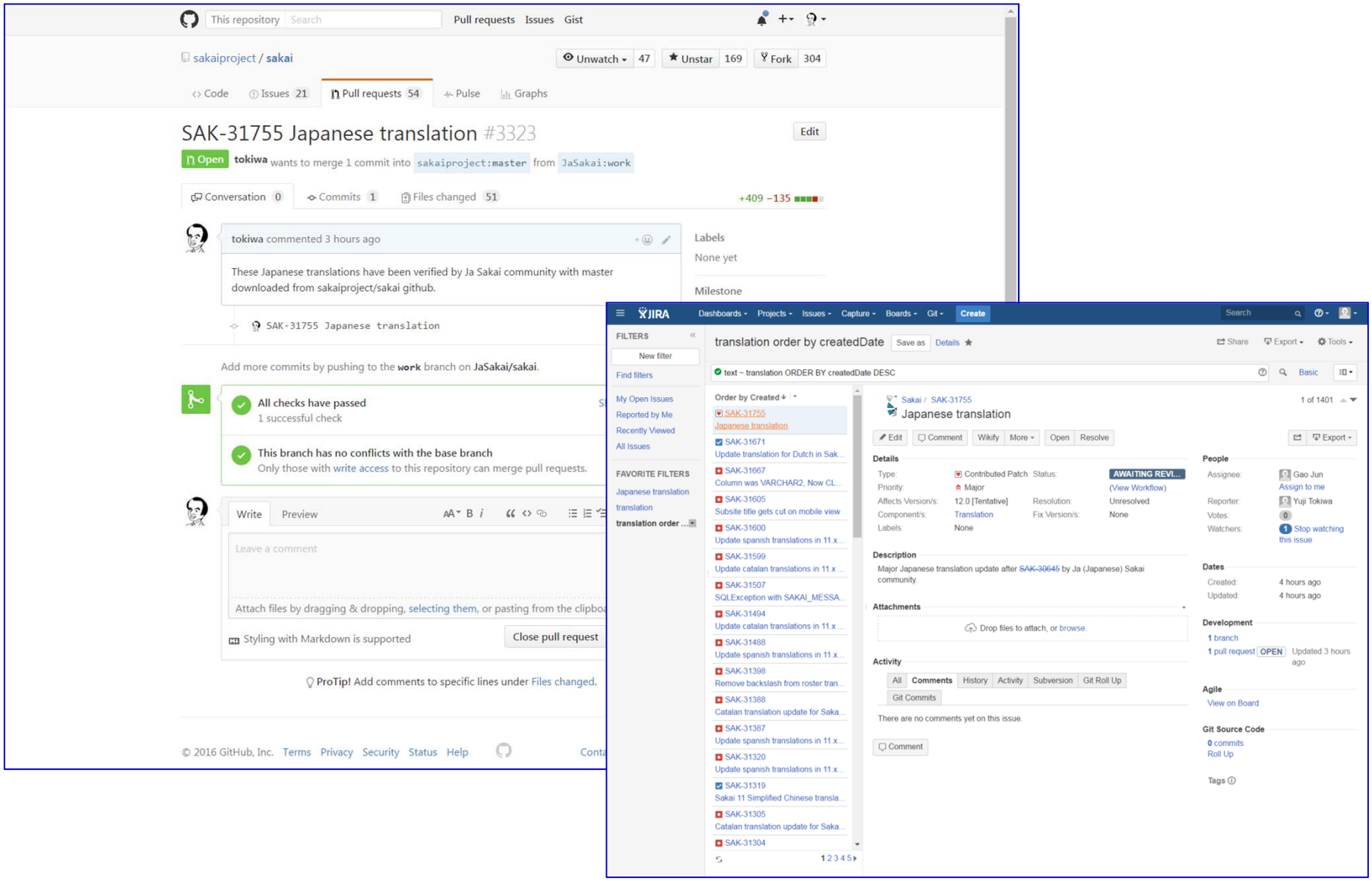
  export JAVA_OPTS="-Dhudson.util.ProcessTree.disable=true“
- JenkinsはジョブごとにWorkspaceが与えられるが、翻訳の適用の有無などで同じWorkspaceを参照するためには、Shared Workspaceプラグインを追加すればよい。

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



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 にて作成する。git checkout –b work origin/master
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。