

# 神情協 先端技術研究会 「使えるJenkinsを構築する」 第2回

---

株式会社ジェイエスピー

# 本日の予定

---

1. はじめに
2. 前回について
3. Jenkinsに触れる 第2弾
4. MagicPodについて
5. MagicPodに触れる
6. 成果物について
7. 次回以降について

以下のアカウントを  
確認してください！

- Jenkins
- MagicPod
- GitHub
- Chatwork

# 今後の予定

	日程	内容	形式
第1回	10/10	CIについて講義／Jenkinsに触れる	オンライン（Zoom）
第2回	11/14	プラグイン／MagicPodの使用	オンライン（Zoom）
第3回	12/12	MagicPodとの連携／グループ話し合い	オンライン（Zoom）
第4回	1/16	Jenkinsのその他設定／グループ話し合い	オンライン（Zoom）
第5回	2/13	成果発表／懇親会	弊社会議室

■ 時間：10:00～12:00

■ 形式について

新型コロナウイルスの影響で第4回までオンラインとする予定です。  
第5回についても、  
皆様のご要望によりオンラインとする可能性があります。

# 前回について

---

## <目標>

- ① CI と Jenkins について知る
- ② Jenkins で簡単な自動テストをする

# 前回について

---

CIとは・・・ソフトウェア開発において以下を実現すること

1. 定期的なビルドとテスト
2. バグの早期発見と修正
3. 開発の品質と生産性の向上

→ メリットもあるが、デメリットもある（自動テストには注意！）

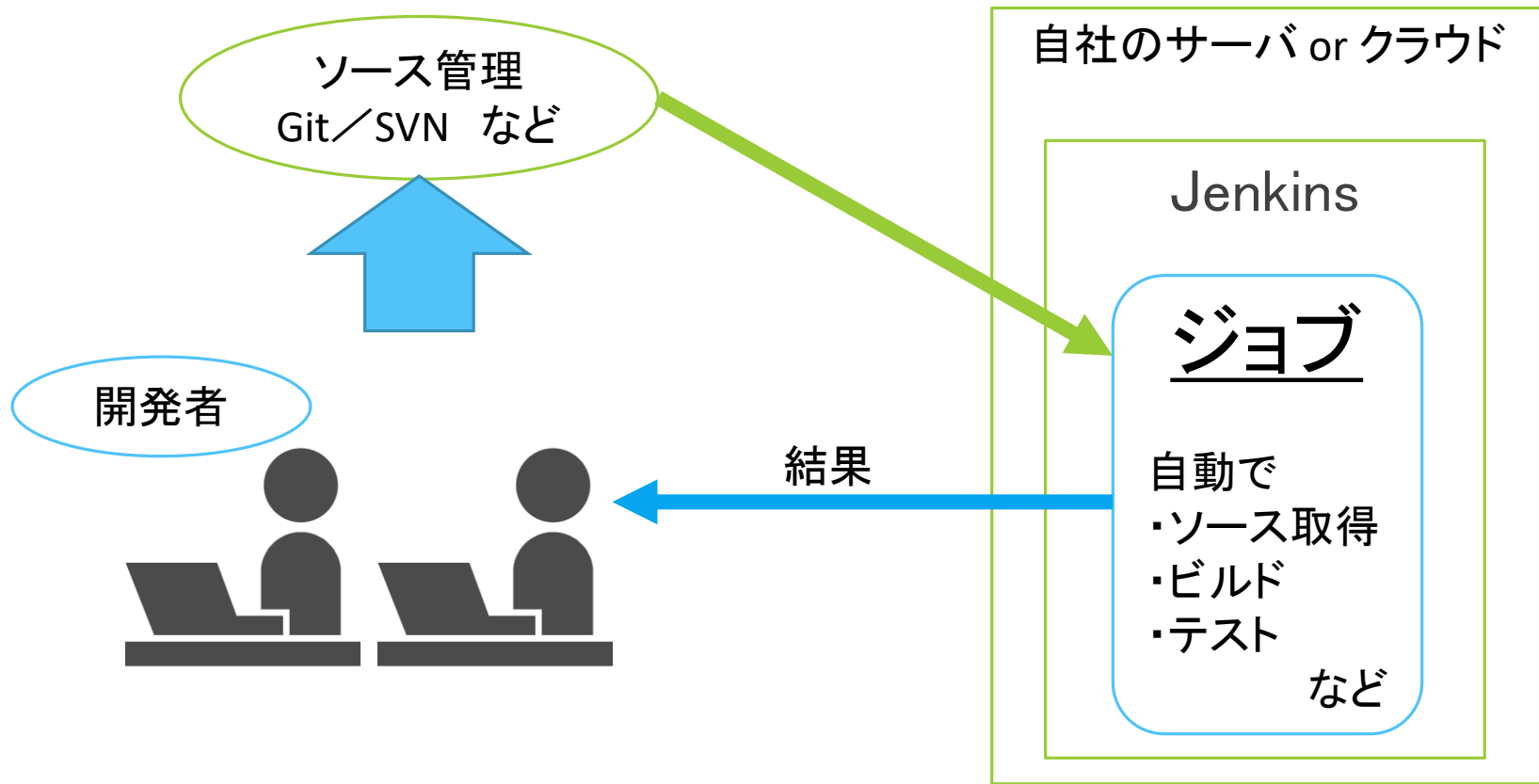
「Jenkins」がこのCIを実現する！

→ 多くのツールと連携が可能。

サーバを立てると、プロジェクト単位で利用が可能。

実際に多くの場で使われている。

# 前回について



# Jenkins Day Japan 2020

---

- Jenkins の最新情報や使い方、今後の展望について聞ける
- 今年はオンラインでの開催
- Jenkinsの前身Hudsonの開発者 川口耕介氏 も登壇
- 参加するには事前予約制なので以下のURLから申し込み

<URL> <https://cloudbees.techmatrix.jp/jenkins-day-japan2020/>



<https://cloudbees.techmatrix.jp/jenkins-day-japan2020/> より引用

# 本日の予定

---

1. はじめに
2. 前回について
3. Jenkinsに触れる 第2弾
4. MagicPodについて
5. MagicPodに触れる
6. 成果物について
7. 次回以降について



# 第2回について

---

## <目標>

- ① Jenkins でほかのツールと連携する。
- ② MagicPod で画面の自動テストをする。

キーワード : Checkstyle、FindBugs、MagicPod

# Jenkins に触ってみる

---

## 前回

1. ジョブの作成
2. GitHubからソースを取得する
3. Ant でビルドをする
4. JUnit でテストをする

## 今回

5. プライベートリポジトリを作る
6. ワークスペースを変える
7. GitHubからソースを取得する
8. Checkstyle で静的解析を行う

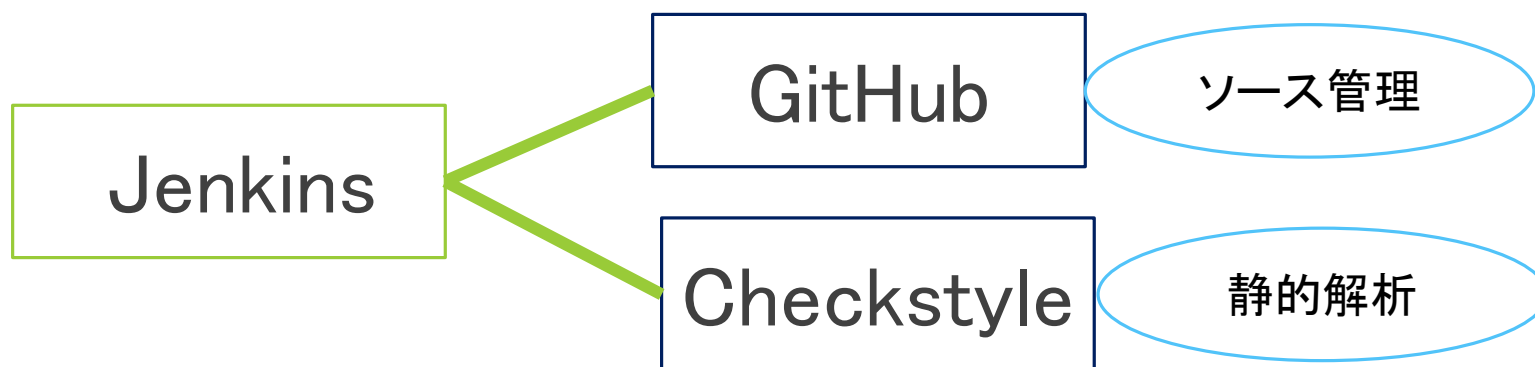
このマークのところまで来たら、  
「手を挙げる」を押してください」↓

ここまで終わったら  
「手を挙げる」

# これのできること

---

プラグインで連携可能！  
より開発環境に近い形に！



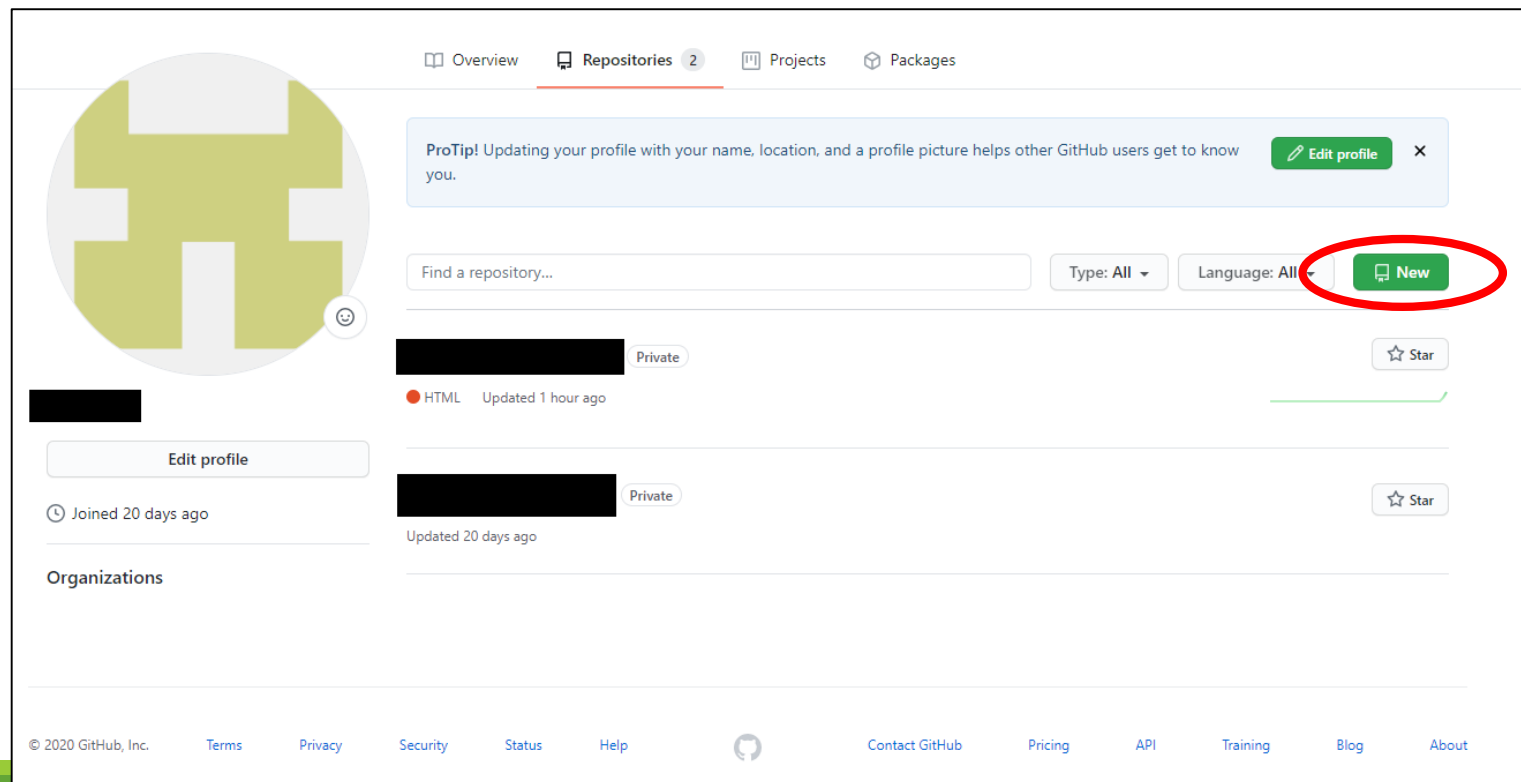
# プライベートリポジトリを作る

---

- 前はパスワードが不要で、GitHubから取得した。  
→ 「パブリック」なリポジトリだったから
- これでは開発してるものが、外から見られる。
- そこで...  
「プライベート」なリポジトリを作成し、開発物を管理する。

# リポジトリを作成①

- ① <https://github.com> にアクセスし、ログインします。
- ② 画面右にある「New」を押す。



# リポジトリを作成②

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?  
[Import a repository.](#)

Owner \* h-senda / Repository name \* sample-repository ✓

Great repository names are short and memorable. Need inspiration? How

Description (optional)

☐ Public  
Anyone on the internet can see this repository. You choose who can commit.

☒ Private  
You choose who can see and commit to this repository.

Initialize this repository with:  
Skip this step if you're importing an existing repository.

☒ Add a README file  
This is where you can write a long description for your project. [Learn more.](#)

☐ Add .gitignore  
Choose which files not to track from a list of templates. [Learn more.](#)

☐ Choose a license  
A license tells others what they can and can't do with your code. [Learn more.](#)

This will set `main` as the default branch. Change the default name in your [settings](#).

Create repository

①

リポジトリ名を入れる  
例: sample-repository

②

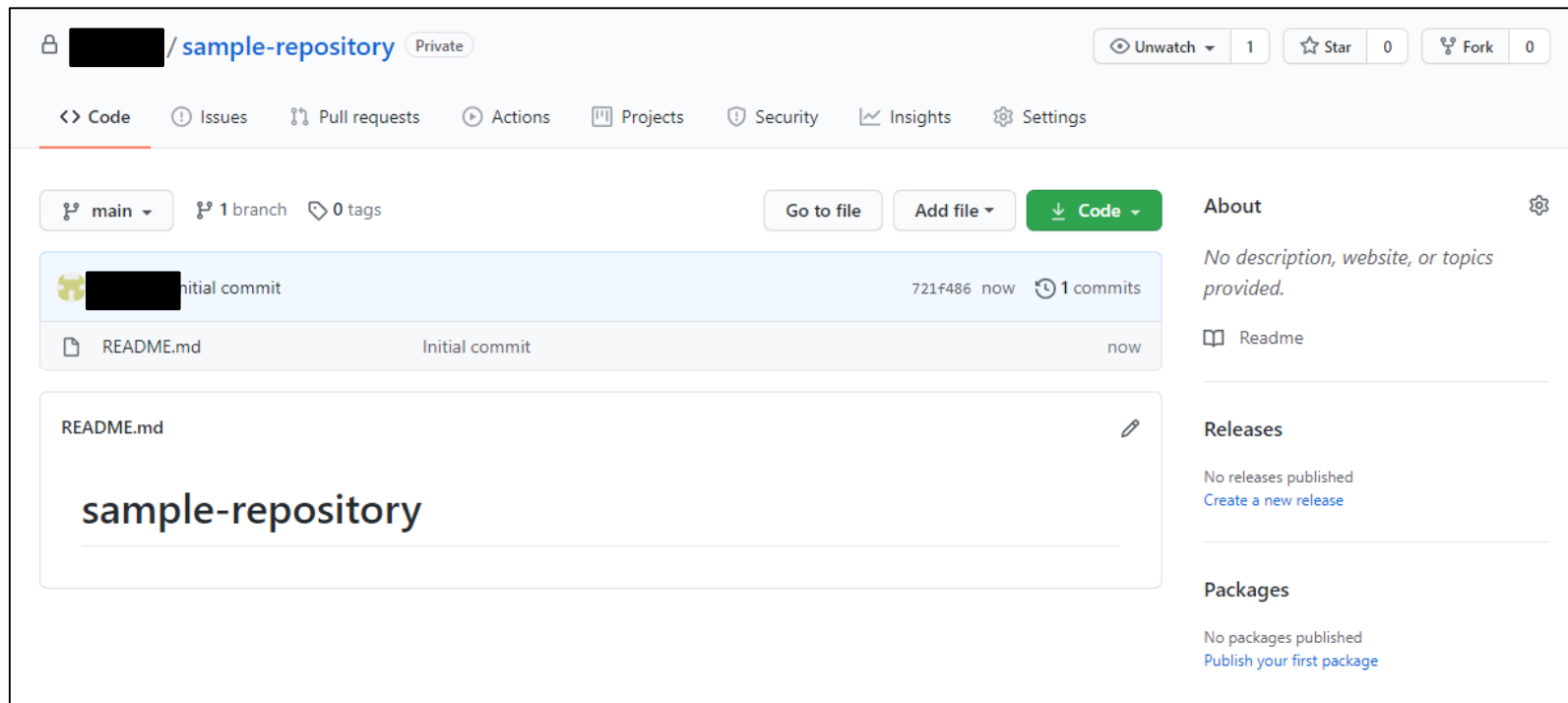
公開か非公開の選択  
「非公開」を選ぶ

③

リポジトリの説明を書くため  
Readmeファイルを入れる

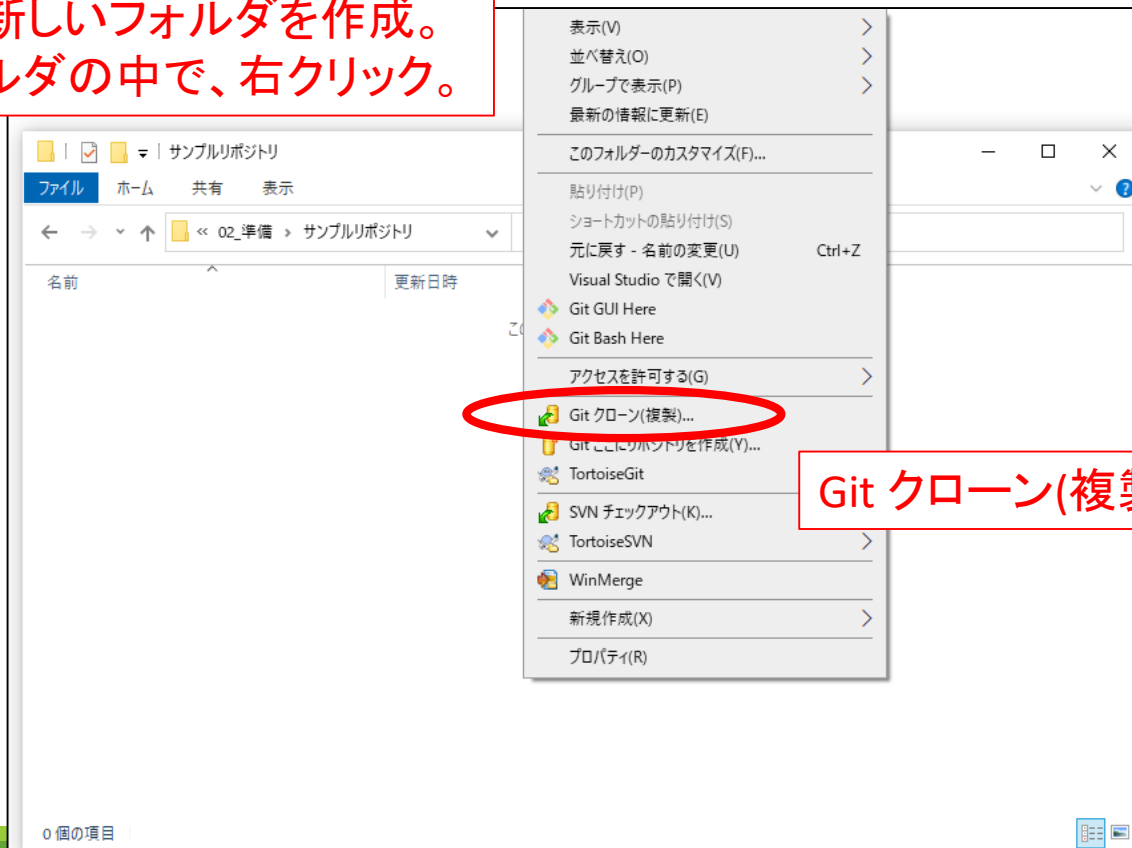
# リポジトリの作成③

作成完了！



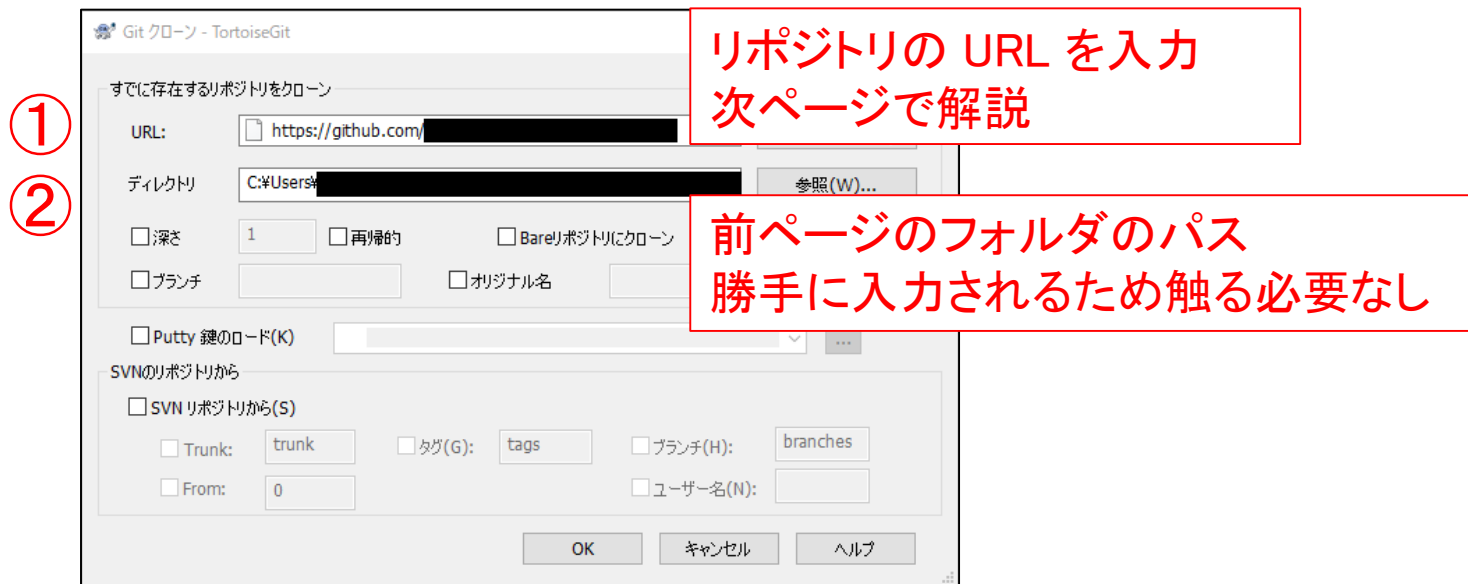
# リポジトリをクローン

エクスプローラーを開き、  
どこかに新しいフォルダを作成。  
そのフォルダの中で、右クリック。





# リポジトリのクローン



# リポジトリのクローン

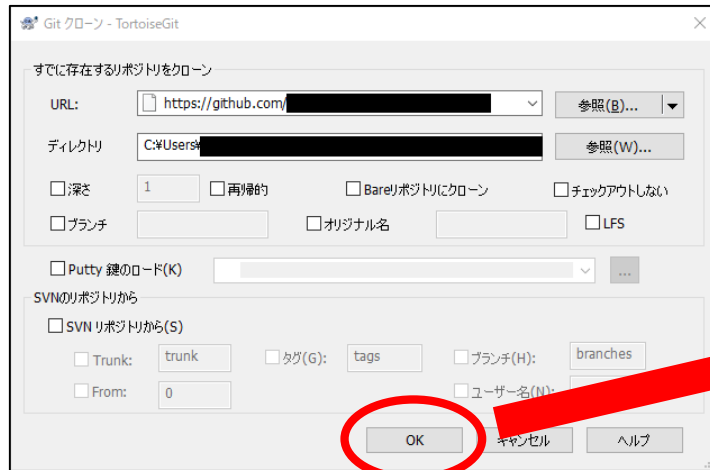
リポジトリの URL の取得方法  
先程作ったリポジトリのページ

The screenshot shows the GitHub interface for a repository named 'sample-repository'. The repository is private and has 1 branch (main). The 'Code' button is highlighted with a red circle and labeled '① 緑の「Code」を押す'. Below the repository name, the 'Clone' button is highlighted with a red circle and labeled '② 箱のようなボタンを押す URL コピーができる'. The 'Clone' button is a green button with a white icon of a box. The 'Clone' button is also highlighted with a red circle and labeled '③ 緑の「Code」を押す'.

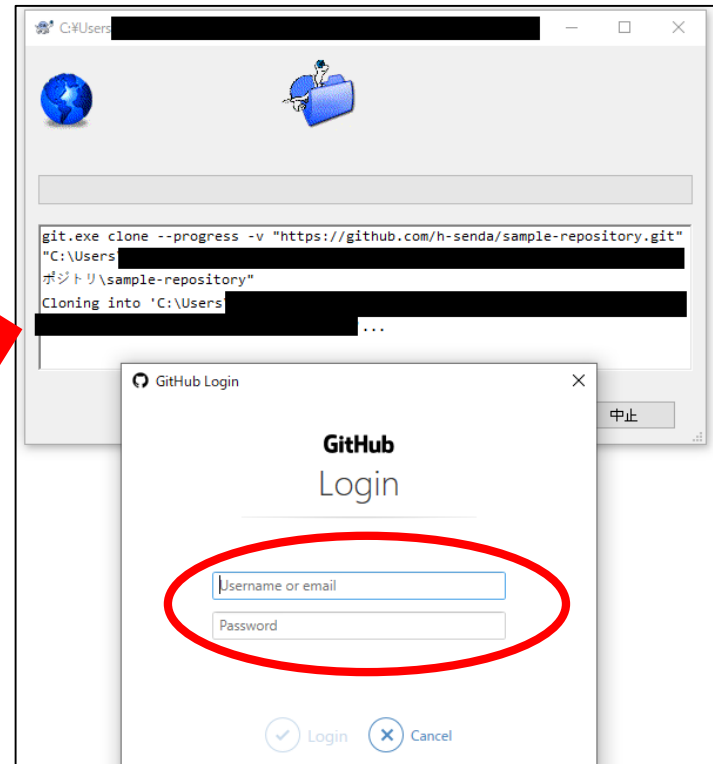
① 緑の「Code」を押す

② 箱のようなボタンを押す  
URL コピーができる

# リポジトリのクローン



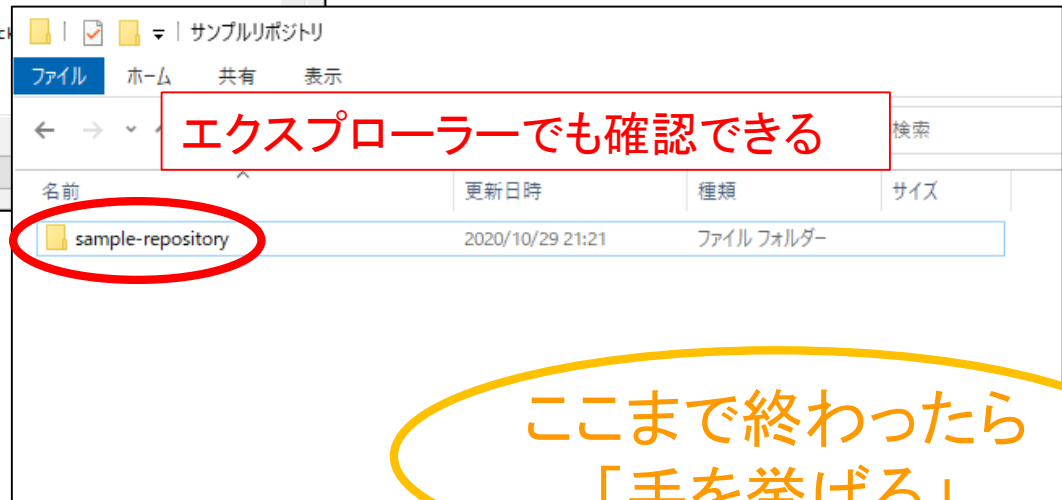
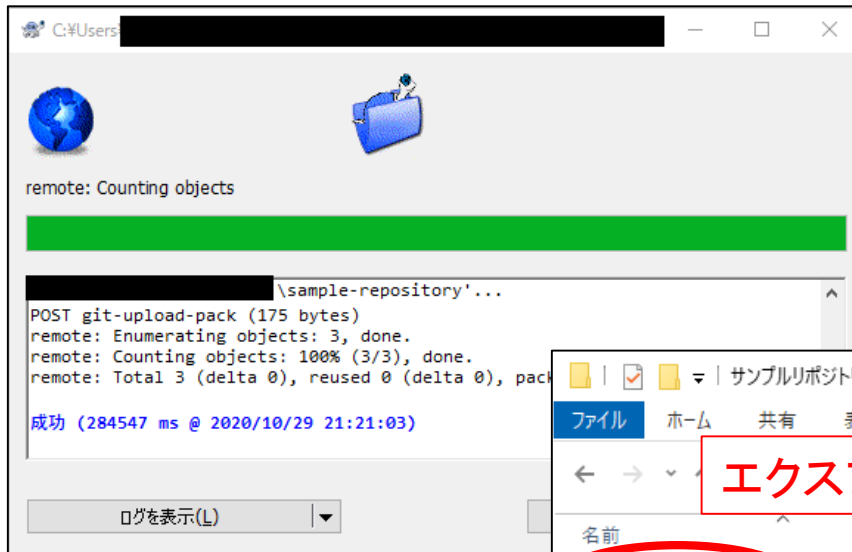
OKボタンを押し、クローン開始



GitHub のログインを求められるので、作成したアカウントでログインをする

# リポジトリのクローン

クローン完了



ここまで終わったら  
「手を挙げる」

# サンプルコードをプッシュ

サンプルコードを作成したりポジトリにプッシュします。

すでに第1回の Jenkins でコードを取得しているので、コピーをします。

取得したコードのパスはコンソールの一番上に書いてある！



The screenshot shows the Jenkins web interface. On the left is a sidebar with navigation links: 'プロジェクトへ戻る', '状態', '変更履歴', 'コンソール出力' (highlighted), 'ブレンテキスト表示', '説明の編集', and 'Delete build '#1''. The main area is titled 'コンソール出力' (Console Output). It shows the execution of a build by user 'hayato\_senda'. The console output is as follows:

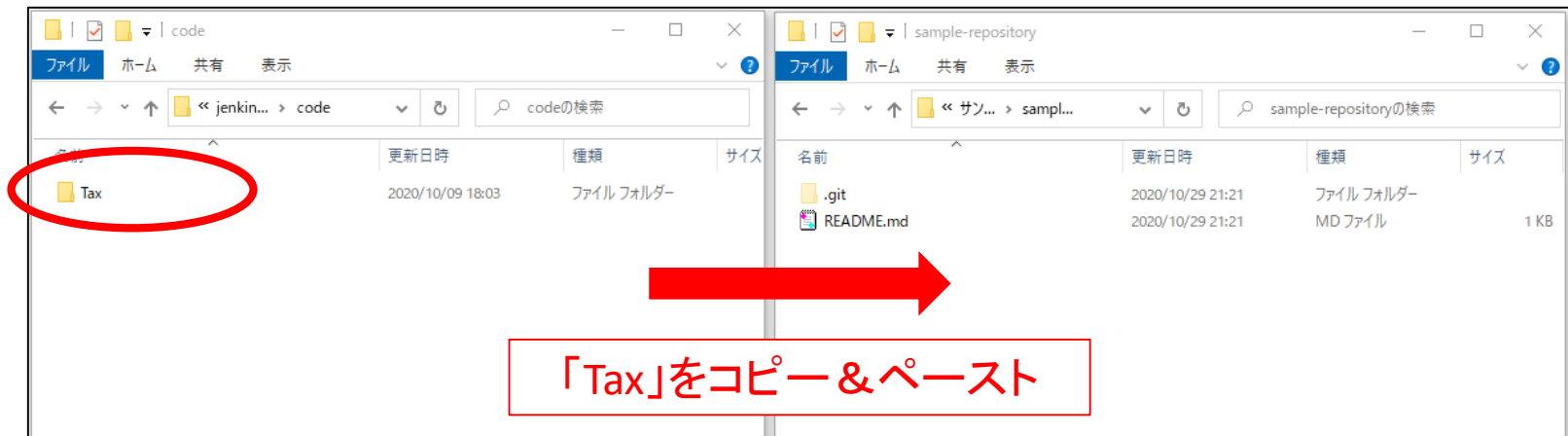
```
Running as SYSTEM
ビルドします。 ワークスペース: C:\Windows\system32\config\systemprofile\AppData\Local\Jenkins.jenkins\workspace\SampleJob
[SampleJob] $ cmd /c echo "Hello World"
C:\Windows\system32\config\systemprofile\AppData\Local\Jenkins.jenkins\workspace\SampleJob>echo "Hello World"
"Hello World"
C:\Windows\system32\config\systemprofile\AppData\Local\Jenkins.jenkins\workspace\SampleJob>exit 0
Finished: SUCCESS
```

A red oval highlights the first line of the console output: 'ビルドします。 ワークスペース: C:\Windows\system32\config\systemprofile\AppData\Local\Jenkins.jenkins\workspace\SampleJob'.

# サンプルコードをプッシュ

Jenkinsで取得したフォルダ  
[ジョブのパス]¥jenkins-study-group¥code

先程クローンしたリポジトリ

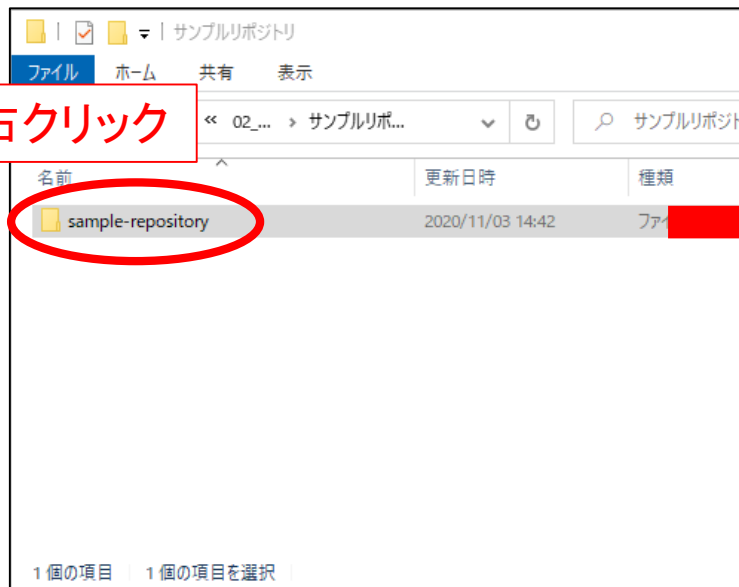


# サンプルコードをプッシュ

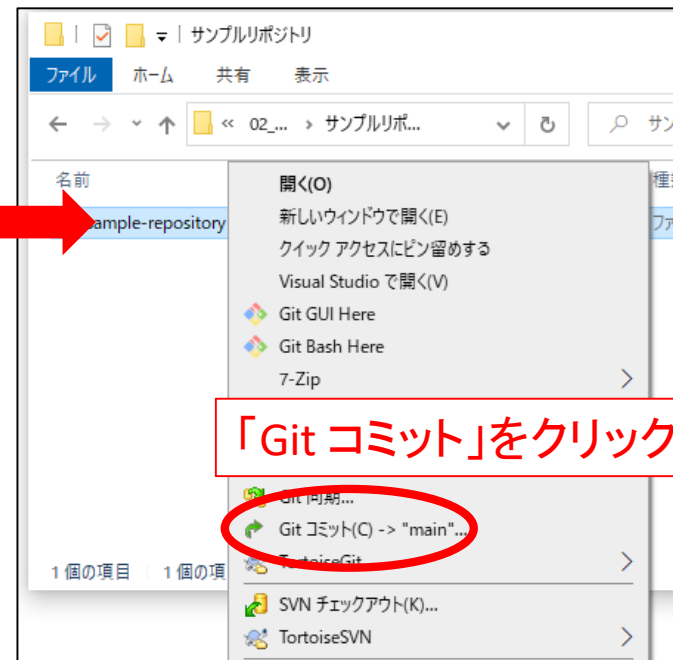
TortoiseGit をインストールした人は...

先程クローンしたリポジトリ  
「Tax」をコピーした後

右クリック



「Git コミット」をクリック



# サンプルコードをプッシュ

TortoiseGit をインストールした人は...

The screenshot shows the TortoiseGit commit dialog box. It has a title bar with the path 'C:\Users\... \sample-repository ~...'. The 'Commit to:' field is set to 'main'. There are checkboxes for 'Commit to: main' and '新しいブランチ' (New branch). The 'Commit message (M):' field is empty. Below this, there are checkboxes for '最後のコミットをやり直し(L)' (Reset last commit), '著述日時を設定する(D)' (Set author and date), and '作者を設定(T)' (Set author). The 'Commit message' field is empty. Below this, there are checkboxes for '変更した項目をすべてチェック' (Check all changed items), 'すべて(A)' (All), '差し(M)' (Staged), and 'バージョン管理下のファイルのみ' (Only files under version control). The 'すべて(A)' checkbox is circled in red. Below this, there is a list of files with checkboxes. The files are: Tax/ant/checkstyle/checkstyle-5.5-all.jar, Tax/ant/checkstyle/sun\_checks.xml, Tax/ant/findbugs/README.txt, Tax/ant/findbugs/bin/addMessages, Tax/ant/findbugs/bin/computeBugHistory, Tax/ant/findbugs/bin/convertXmlToText, Tax/ant/findbugs/bin/copyBuggySource, Tax/ant/findbugs/bin/defectDensity, and Tax/ant/findbugs/bin/defectDensity. The checkboxes for all these files are checked. Below the file list, there are checkboxes for 'バージョン管理外のファイルを表示(U)' (Show files not under version control) and 'サブモジュールを自動的に選択しない' (Do not automatically select submodules). The 'バージョン管理外のファイルを表示(U)' checkbox is checked. Below this, there are checkboxes for 'プロジェクト全体を表示(W)' (Show all projects) and 'メッセージのみ(Y)' (Only messages). The 'プロジェクト全体を表示(W)' checkbox is checked. At the bottom, there is a 'Commit(O)' button, which is circled in red. To the right of the 'Commit(O)' button is a 'キャンセル' (Cancel) button. There is also a 'パッチを表示>>' (Show patch >>) button.

① メッセージに  
「サンプルコードをコミット」を入力

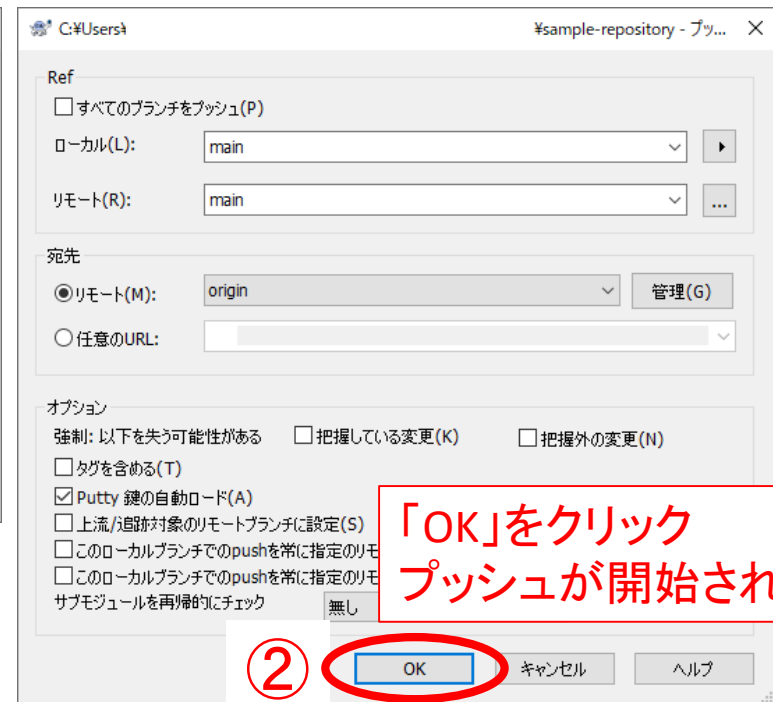
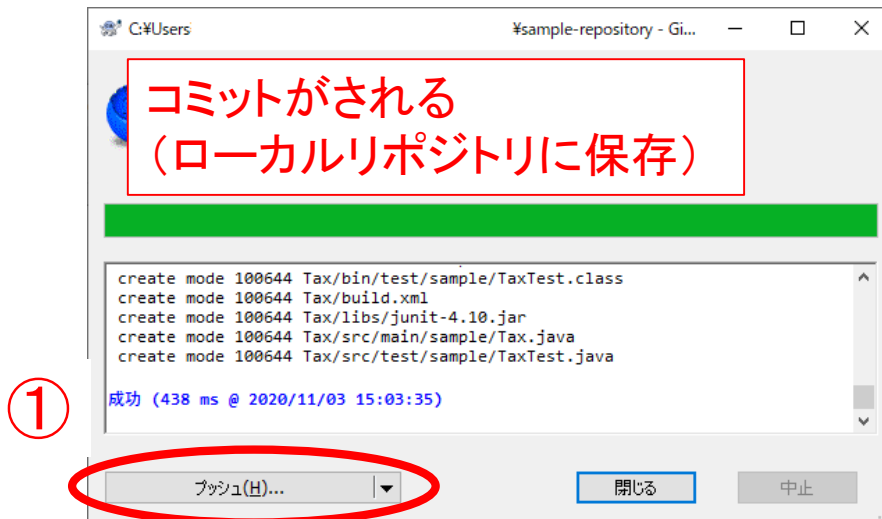
② 「すべて」をクリック  
下のファイルにチェックがつく

③ 「コミット」をクリック



# サンプルコードをプッシュ

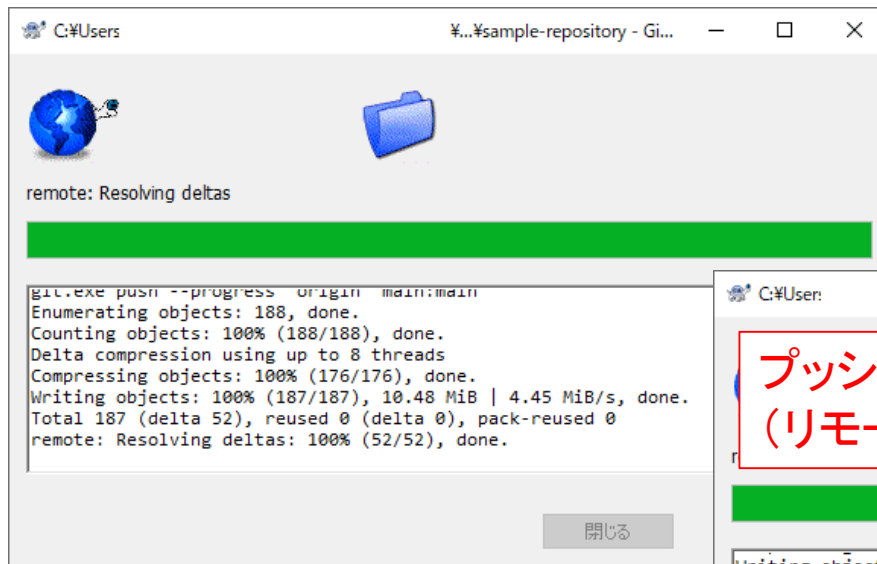
TortoiseGit をインストールした人は...



「OK」をクリック  
プッシュが開始される

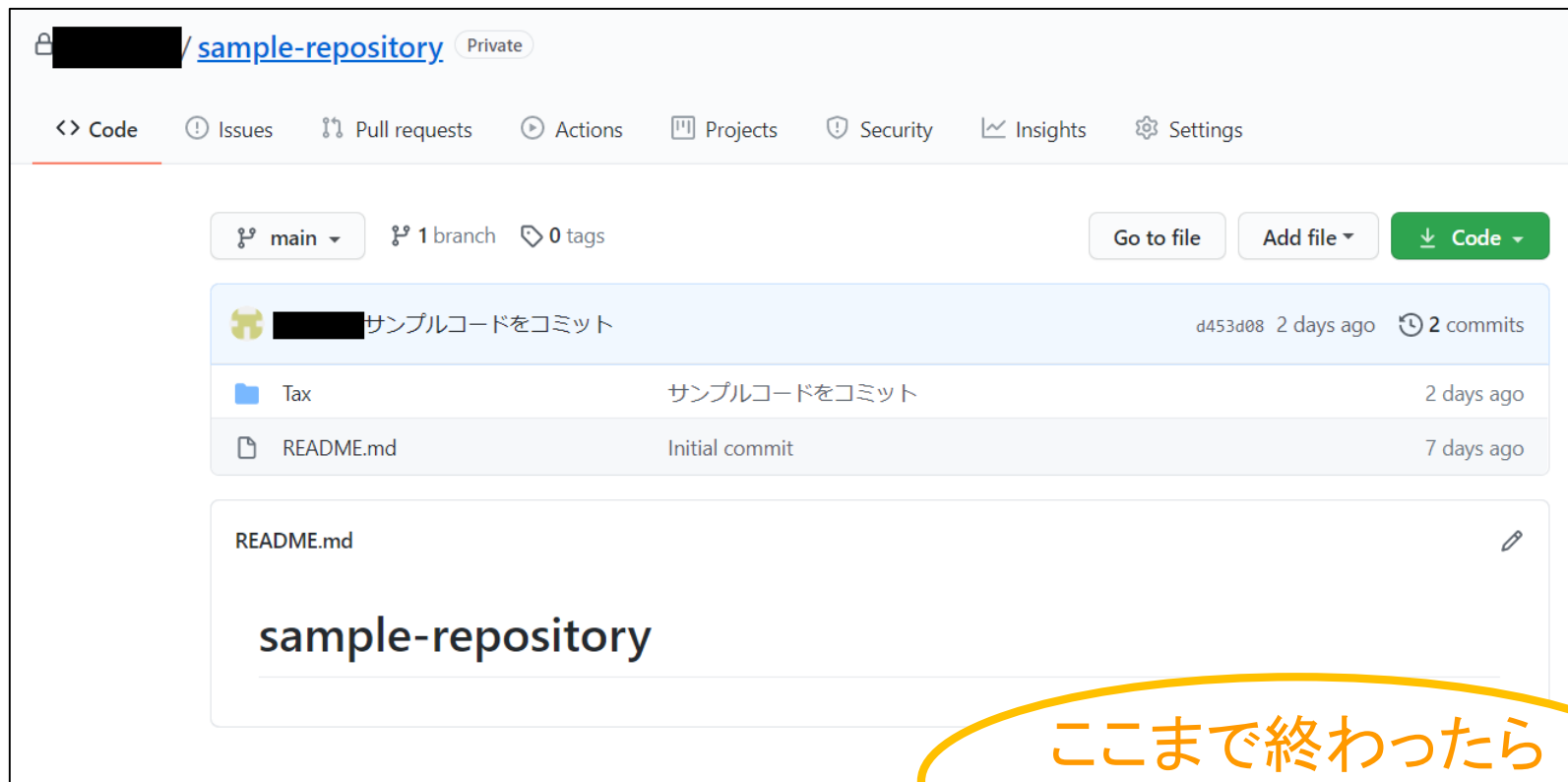
# サンプルコードをプッシュ

TortoiseGit をインストールした人は...



# サンプルコードをプッシュ

GitHub の画面を見るとプッシュしたものが見れる



ここまで終わったら  
「手を挙げる」

# Jenkins に触ってみる

---

## 前回

1. ジョブの作成
2. GitHubからソースを取得する
3. Ant でビルドをする
4. JUnit でテストをする

## 今回

5. プライベートリポジトリを作る
6. ワークスペースを変える
7. GitHubからソースを取得する
8. Checkstyle で静的解析を行う

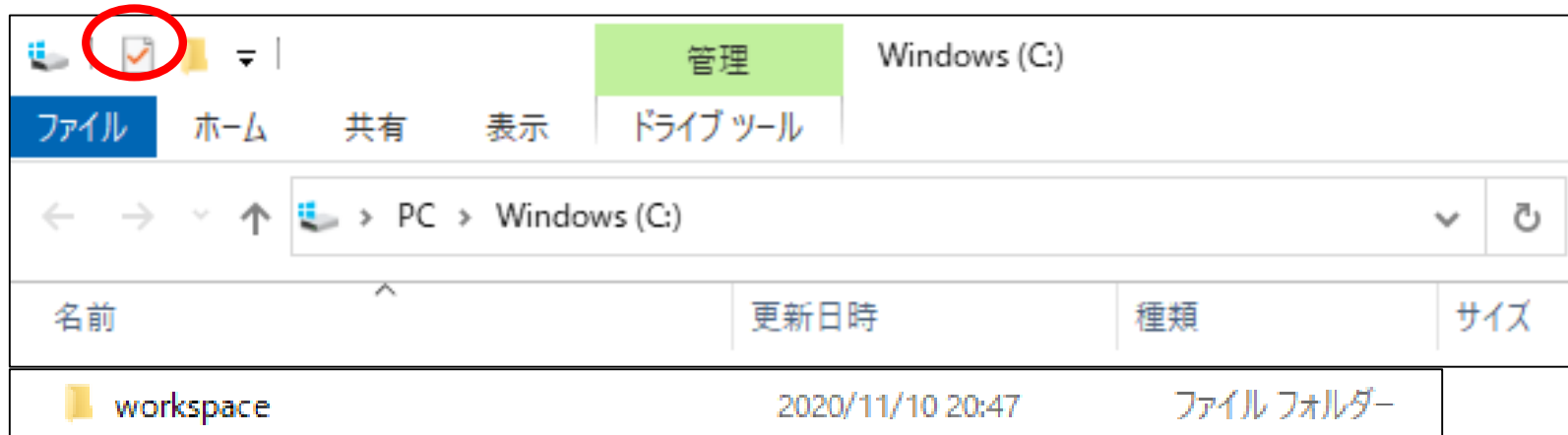
# ワークスペースを変える

---

- ワークスペースとは...  
ジョブで取得／作成するファイルを保存するフォルダのこと
- 新しくジョブを作成し、ファイルの保存先を変更する。
- ファイルを更新する上で、権限により更新できない時がある。  
→ それを防ぎたい！

# ワークスペースを変える

ドライブCの直下に「 workspace 」というフォルダを作る



# ワークスペースを変える

Jenkinsのページでジョブをコピーする

①



The screenshot shows the Jenkins dashboard. The '新規ジョブ作成' (Create New Job) button in the left sidebar is circled in red.

②

ジョブ名 (前回作ったものとは異なる名前で)



The screenshot shows the 'Enter an item name' dialog. The 'SampleJob' text in the input field is circled in red.

③

第1回で作ったジョブの名前を入力



The screenshot shows the 'Copy from' dialog. The 'Copy from' text in the input field is circled in red.

④



The screenshot shows the 'OK' button in the dialog, which is circled in red.

# ワークスペースを変える

ジョブの設定を行う





# ワークスペースを変える

## General の項目



隠れている項目を表示するために、  
「高度な設定」を押す ↓

# ワークスペースを変える

## ワークスペースの設定を行う

① ☒ カスタムワークスペースを使用

② ディレクトリ

表示用プロジェクト名

☐ 依存している上流プロジェクトをビルドする

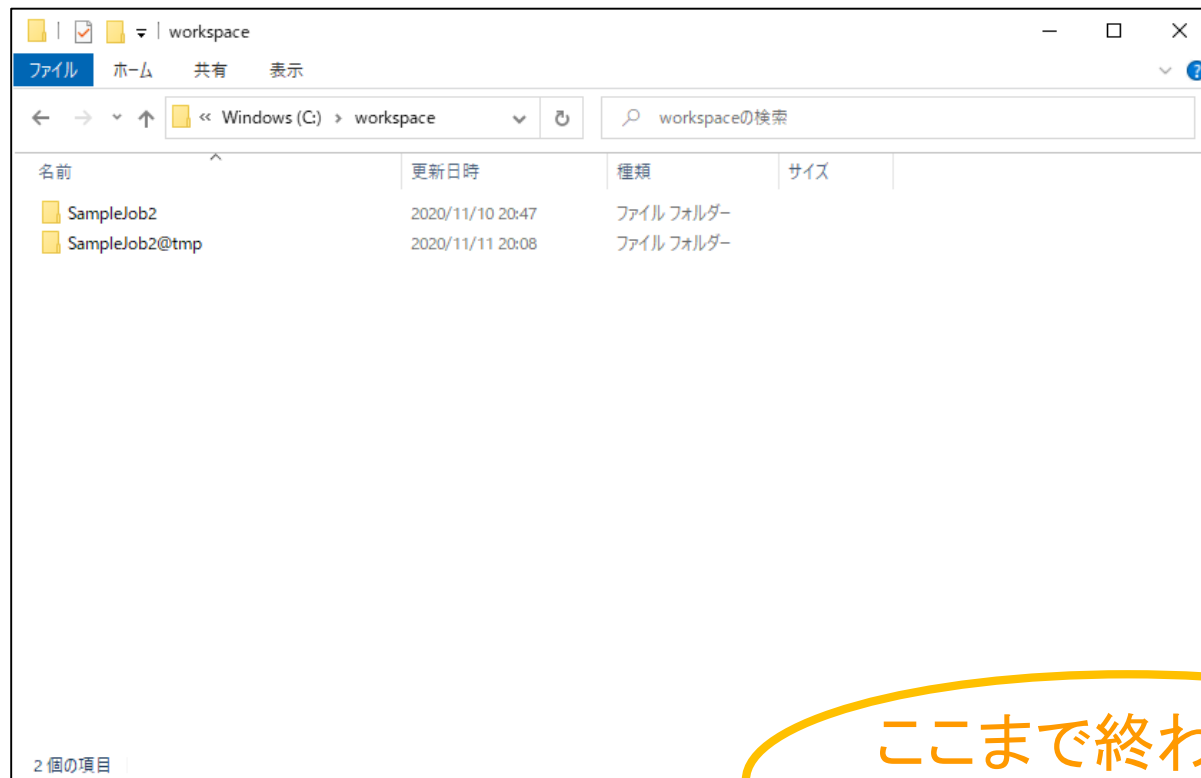
↑Cドライブ直下に「workspace」を作成し、  
「C:¥workspace¥SampleJob2」と入力する

保存 Apply

保存し、ビルドを実行する

# ワークスペースを変える

実際にフォルダができているかを確認



ここまで終わったら  
「手を挙げる」

# Jenkins に触ってみる

---

## 前回

1. ジョブの作成
2. GitHubからソースを取得する
3. Ant でビルドをする
4. JUnit でテストをする

## 今回

5. プライベートリポジトリを作る
6. ワークスペースを変える
7. GitHubからソースを取得する
8. Checkstyle で静的解析を行う

# GitHubからソースを取得する

- 「プライベート」なりポジトリからソースを取得するには...  
GitHub アカウントの情報が必要！
- Jenkins でアカウント情報を登録し、取得できるように設定する。

↓前回は「パブリック」なりポジトリだったため、  
認証情報に何も入れなくても取得できた。



リポジトリ

リポジトリURL

認証情報 - なし 追加

高度な設定...

リポジトリの追加

# GitHubからソースを取得する

前回と同様にURLをコピーする

The screenshot shows a GitHub repository page for a user named 'sample-repository'. The repository is private. The 'Code' button is circled in red, with a red circle and the number '1' next to it. Below the 'Code' button, the 'Clone' dropdown menu is open, showing the 'HTTPS' option selected. The clone URL is circled in red, with a red circle and the number '2' next to it. A red box with the text 'このリポジトリのURLをコピーする' (Copy the URL of this repository) is overlaid on the clone URL.

①

②

このリポジトリのURLをコピーする

# GitHubからソースを取得する

## ジョブ設定の「ソースコード管理」の項目

**ソースコード管理**

☐ なし  
☒ Git

リポジトリ

リポジトリURI ① https://github.com/[redacted]/sample-repository.git ?

認証情報 - なし ② 追加 ▲ ?

③ Jenkins 高度な設定...

リポジトリの追加


ビルドするブランチ


ブランチ指定子 (空欄はすべてを指定) X \*/master ?

ブランチの追加

リポジトリ・ブラウザ (自動) ?

# GitHubからソースを取得する

 **Jenkins Credentials Provider: Jenkins**

 **認証情報の追加**

Domain

グローバルドメイン

種類

ユーザー名とパスワード

スコープ

グローバル

①

ユーザー名

GitHub アカウントのアカウント名

②

パスワード

GitHub アカウントのパスワード

③

ID

「sampleGit」(この認証情報のID)

説明

④

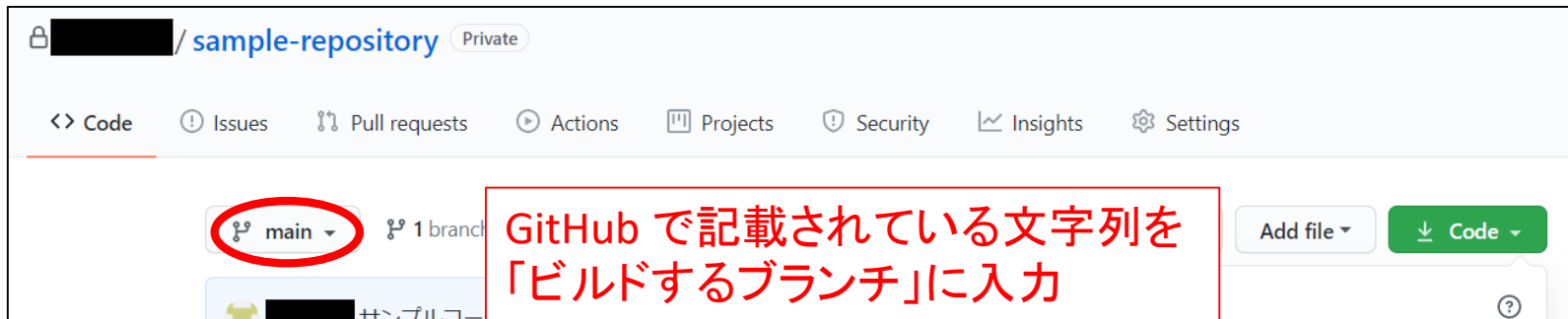
追加

中止



# GitHubからソースを取得する

## ブランチの指定



# GitHubからソースを取得する

リポジトリ

リポジトリURL

認証情報

ビルドするブランチ

ブランチ指定子 (空欄はすべてを指定)

リポジトリ・ブラウザ (自動)

追加処理

保存し、ビルドを実行しましょう！

ここまで終わったら  
「手を挙げる」

# Jenkins に触ってみる

---

## 前回

1. ジョブの作成
2. GitHubからソースを取得する
3. Ant でビルドをする
4. JUnit でテストをする

## 今回

5. プライベートリポジトリを作る
6. ワークスペースを変える
7. GitHubからソースを取得する
8. Checkstyle で静的解析を行う

# Checkstyle で静的解析を行う

- Checkstyle は Java のソースを解析するツール。
- コーディング規約に沿っているかを解析する。
- 例えば、Java で言うと以下のようなものに警告がでる。

```
public class Tax {  
    public String test;  
  
    public static void main(String[] args) {  
        /* 処理 */  
    }  
  
    public static int calcConsumptionTax(int price) {  
        int res = (int)(price * 0.08);  
        /* 処理 */  
    }  
}
```

Javadoc 用のコメントがない！

マジックナンバーが  
使われている！

# Checkstyle で静的解析を行う

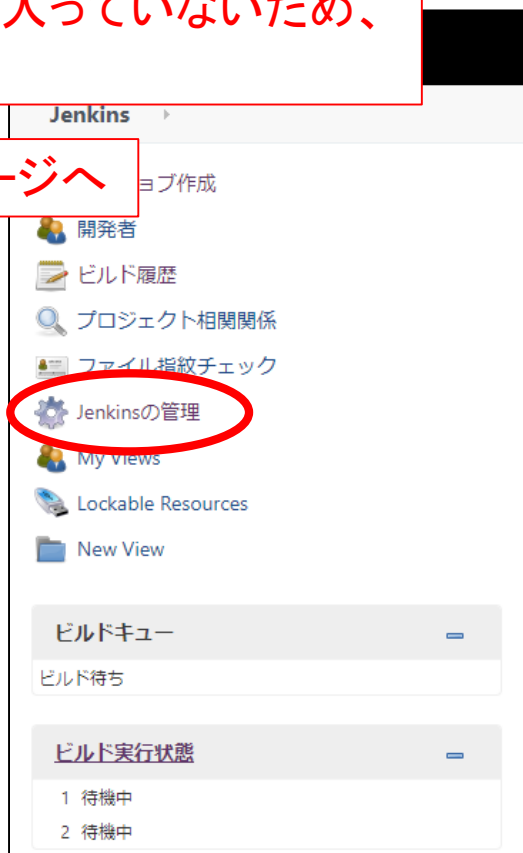
---

- Checkstyle を Jenkins で使用するには「プラグイン」が必要
- 去年までは「**Checkstyle プラグイン**」という専用のプラグインがあった  
→脆弱性を理由に廃止(11月には削除されてた...)
- そこで登場したのが「**Warnings Next Generation プラグイン**」

# Checkstyle で静的解析を行う

デフォルトでプラグインが入っていないため、  
自分で入れる

オフラインの方は 90ページへ



# Checkstyle で静的解析を行う

プラグインを入れる

## System Configuration



### システムの設定

システム全体の振る舞いやパスを設定します。



### Global Tool Configuration

Configure tools, their locations and automatic installers.



### プラグインの管理

Jenkinsの機能を拡張可能なプラグインの追加、削除、無効化および有効化を行います。

🔴 アップデートあり

# Checkstyle で静的解析を行う

今あるプラグインを最新版にする

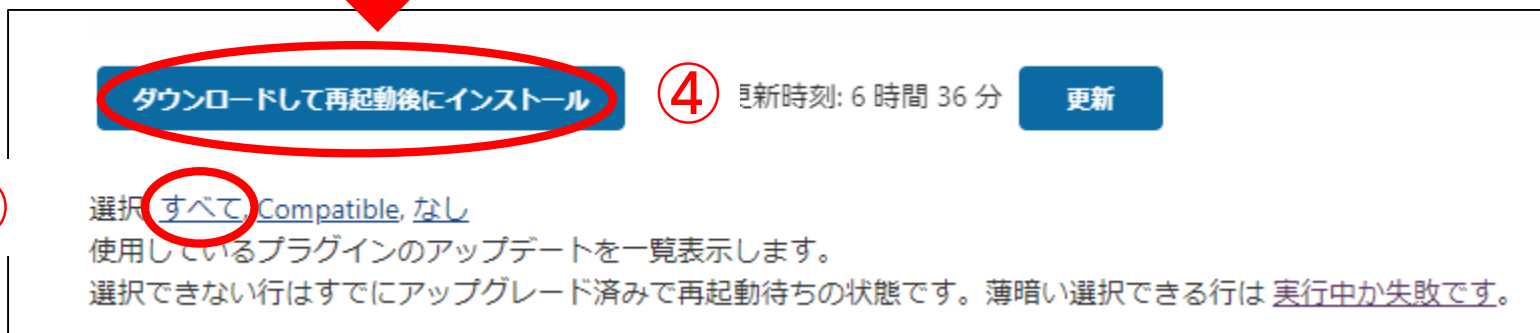
①



②

ページの一番下までスクロール

③



④

最新時刻: 6 時間 36 分

更新

選択 すべて Compatible, なし

使用しているプラグインのアップデートを一覧表示します。

選択できない行はすでにアップグレード済みで再起動待ちの状態です。薄暗い選択できる行は 実行中か失敗です。



# Checkstyle で静的解析を行う

プラグインがアップグレードされるのを待つ

## プラグインのインストール/アップグレード

準備

- インターネットとの接続をチェックします。
- jenkins-ci.orgとの接続をチェックします。
- 成功

LDAP

● 正常にダウンロードしました。次のブート時に有効になります。

SCM API

● 正常にダウンロードしました。次のブート時に有効になります。

Script Security

● 正常にダウンロードしました。次のブート時に有効になります。

「次のブート時に有効になります」とある場合は  
ページ一番下までスクロールし、再起動にチェック

➡ ☐ インストール完了後、ジョブがなければJenkinsを再起動する

# Checkstyle で静的解析を行う

プラグイン「Warnings Next Generation」を入れる

The screenshot shows the Checkstyle plugin installation interface. It includes a search bar at the top, a list of plugins with status buttons (アップデート, 利用可能, インストール済み, 高度な設定), and a table of installed plugins. Four steps are highlighted with red circles and numbers:

- ① The '利用可能' (Available) button for the 'Warnings Next Generation' plugin is circled in red.
- ② The search bar contains the text 'warning next', and a red box with the text '「warning next」と入力' (Enter 'warning next') points to the search bar.
- ③ The checkbox for 'Warnings Next Generation' is checked, and a red box with the text 'チェック' (Check) points to the checkbox.
- ④ The '再起動せずにインストール' (Install without restarting) button is circled in red.






# Checkstyle で静的解析を行う

プラグインがインストールされるので待つ

## プラグインのインストール/アップグレード

準備

- インターネットとの接続をチェックします。
- jenkins-ci.orgとの接続をチェックします。

DataTables.net API	 待機中
Forensics API	 待機中
Analysis Model API	 待機中
Warnings Next Generation	 待機中
Loading plugin extensions	 Pending

➡ [ページの先頭へ戻る](#)  
(すぐにインストールしたプラグインを使用できます)

➡ ☐ インストール完

今回は再起動の必要がない  
終わったら 92ページへ

# Checkstyle で静的解析を行う

---

- DataTables.net API  
<https://updates.jenkins.io/download/plugins/data-tables-api/>
- Forensics API  
<https://updates.jenkins.io/download/plugins/forensics-api/>
- Analysis Model API  
<https://updates.jenkins.io/download/plugins/analysis-model-api/>
- Warnings Next Generation  
<https://updates.jenkins.io/download/plugins/warnings-ng/>

プラグインのインストールは次ページ

# Checkstyle で静的解析を行う

The screenshot shows the Checkstyle configuration interface in an IDE. It is divided into two main sections: 'HTTP Proxyの設定' and 'プラグインのアップロード'.

**Step 1:** In the top bar, the '高度な設定' (Advanced Settings) button is circled in red.

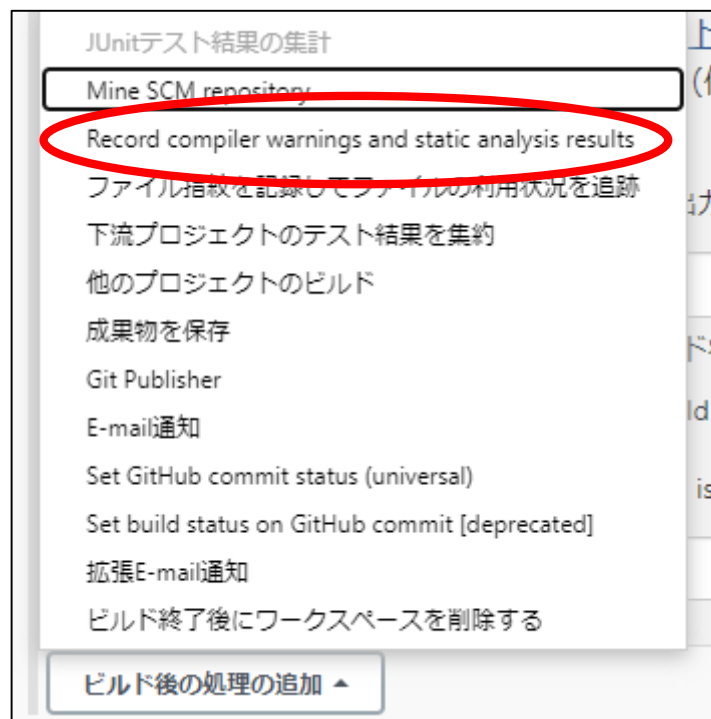
**Step 2:** In the 'プラグインのアップロード' section, the 'ファイルを選択' (Select File) button is circled in red. A red box next to it contains the text '先程ダウンロードしたファイルを選択' (Select the file you just downloaded).

**Step 3:** The 'アップロード' (Upload) button is circled in red.

Additional details: The top bar also contains 'アップデート' (Update), '利用可能' (Available), and 'インストール済み' (Installed). The 'HTTP Proxyの設定' section has a 'サーバー' (Server) label and an empty text field. The 'プラグインのアップロード' section has a description: '.hpiファイルをアップロードして、プラグインリポジトリ以外からプラグインをインストールできます。' (Upload .hpi files to install plugins from sources other than the plugin repository.)

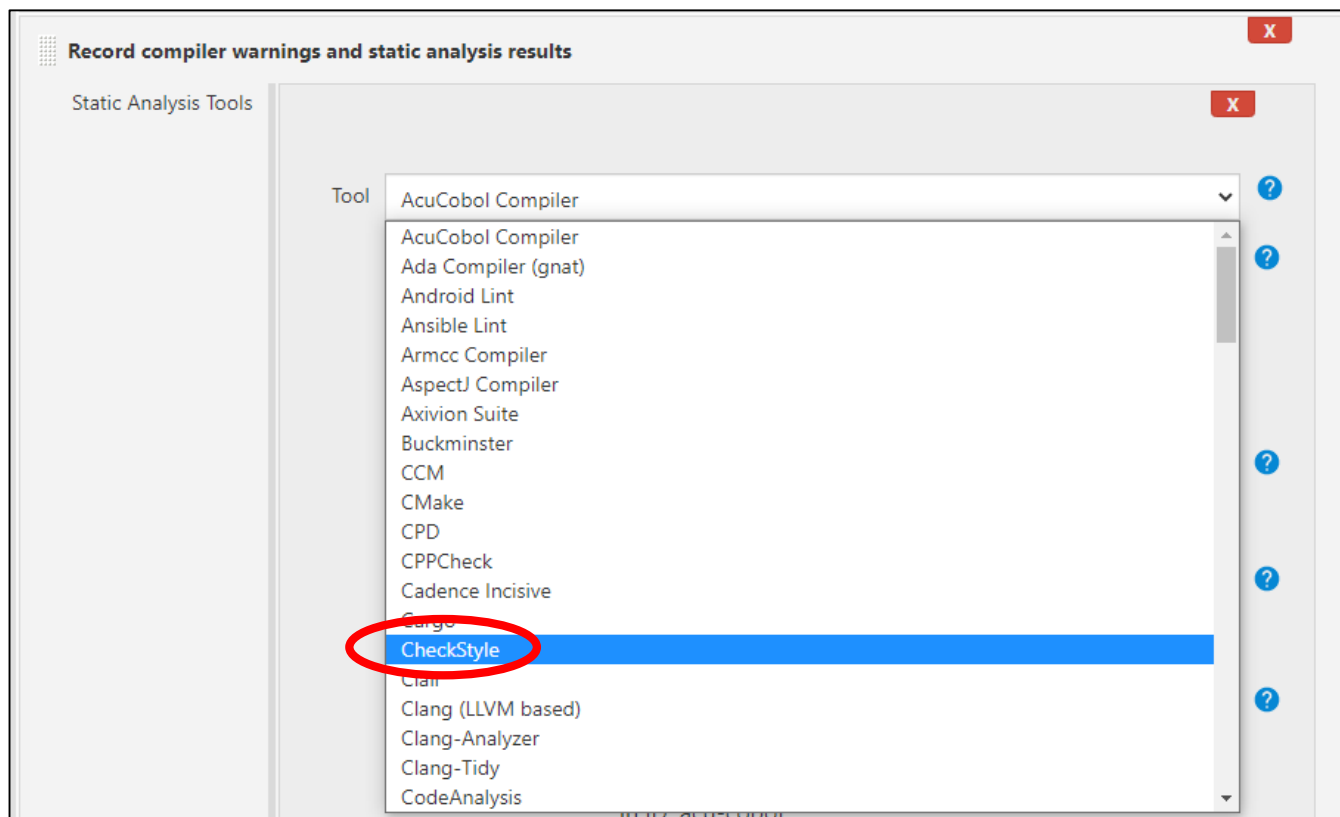
# Checkstyle で静的解析を行う

ジョブ設定の「ビルド後の処理」の項目



# Checkstyle で静的解析を行う

解析ツールを選択



# Checkstyle で静的解析を行う

解析するファイル名を指定

Record compiler warnings and static analysis results

Static Analysis Tools

Tool: CheckStyle

Report File Pattern: Tax\ant\report\\*.xml

Specify the file pattern for the report. If you leave this field empty, then the default file pattern '\*\*/checkstyle-result.xml' will be used.

Skip Symbolic Links: ☐

Toggle whether the file scanner will skip symbolic links.

Report Encoding:



# Checkstyle で静的解析を行う

ビルド実行！



終わったら結果を確認

# Checkstyle で静的解析を行う

解析結果を見る

 ビルド #13 (2020/11/11 19:52:39)

 変更なし

 ユーザー [hayato\\_senda](#) が実行

 リビジョン: 5c78c99dd7160990ec9a4aae44915274ac44eb94

- [refs/remotes/origin/main](#)

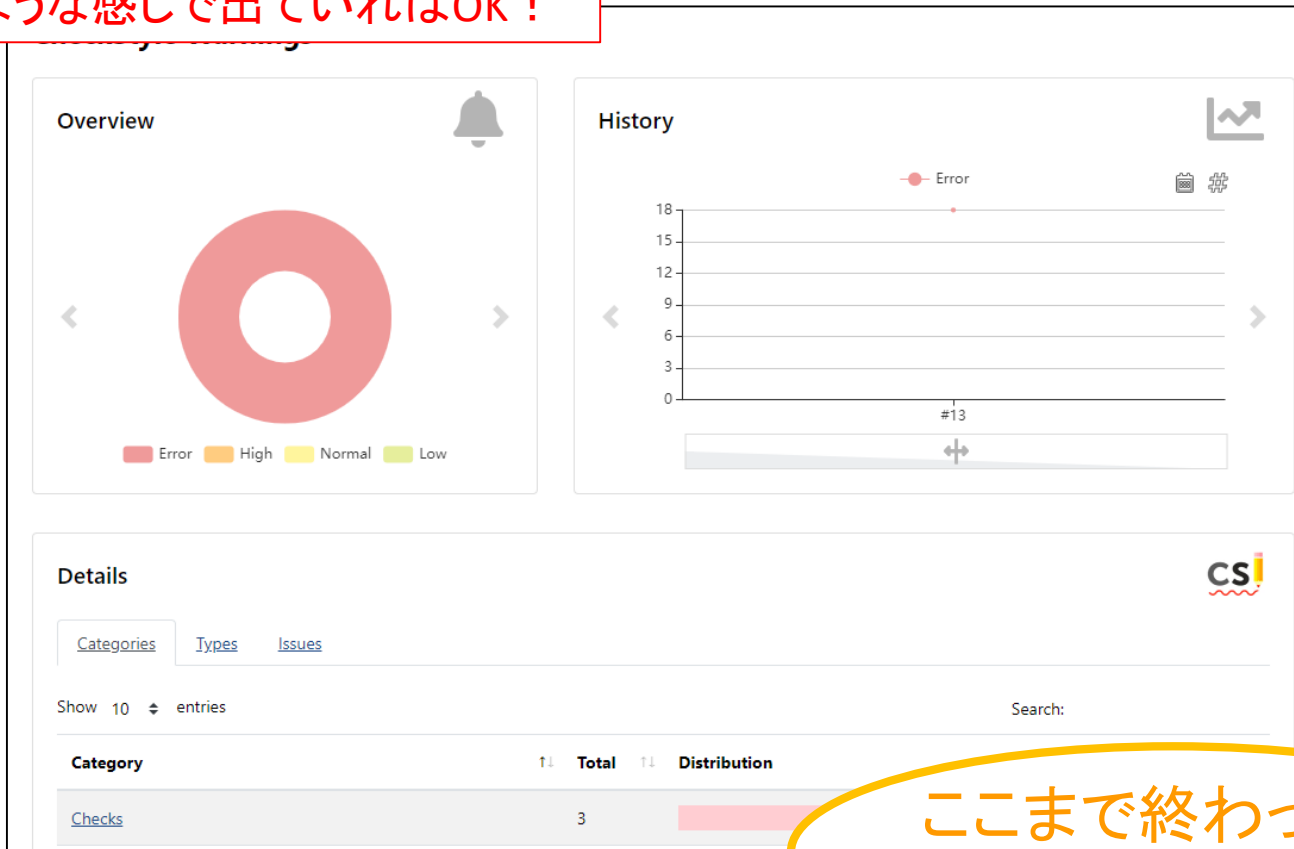
 [テスト結果](#) (1個の失敗 / ±0)

 CheckStyle: [18 warnings](#) 

↓ 解析結果はココ

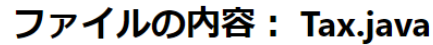
# Checkstyle で静的解析を行う

このような感じで出ていればOK！



ここまで終わったら  
「手を挙げる」

## 何が表示されているか



「マジックナンバーが使われてますよー」  
という警告が表示されている

```
static final int SECONDS_PER_DAY = 24 * 60 * 60;
static final double SPECIAL_RATIO = 4.0 / 3.0;
static final double SPECIAL_SUM = 1 + Math.E;
static final double SPECIAL_DIFFERENCE = 4 - Math.PI;
static final Border STANDARD_BORDER = BorderFactory.createEmptyBorder(3, 3, 3, 3);
static final Integer ANSWER_TO_THE_ULTIMATE_QUESTION_OF_LIFE = new Integer(42);
```

# ほかには...

FindBugs というのもある

The image shows a screenshot of an IDE's 'Add Tool' dialog. The dialog is titled 'Add Tool' and has a red arrow pointing to it from the text '「Add Tool」を押して 解析ツールを追加'. The dialog is divided into two panes. The left pane shows the 'Tool' dropdown set to 'CheckStyle'. The right pane shows the 'Tool' dropdown set to 'FindBugs'. The 'Report File Pattern' field in the right pane contains the text 'Tax\ant\report\\*.xml', which is highlighted by a red box and the text '「Tax¥ant¥report¥\*.xml」と入力'. The 'Add Tool' button at the bottom left is also circled in red.

「Add Tool」を押して  
解析ツールを追加

「FindBugs」を選択

「Tax¥ant¥report¥\*.xml」と入力

# ほかには...

ビルド実行後の結果を見ると...

## ビルド #14 (2020/11/11 20:08:42)



変更なし



ユーザー [hayato\\_senda](#) が実行



リビジョン: [5c78c99dd7160990ec9a4aae44915274ac44eb94](#)

- [refs/remotes/origin/main](#)



[テスト結果](#) (1個の失敗 / ±0)



CheckStyle: [18 warnings](#) ⚠

- Reference build: [SampleJob2 #13](#)




FindBugs: [One warning](#) ⚠

# ほかには...

ビルド実行後の結果を見ると...

「Public で設定する必要はないよねー」という警告

 UuF: 未使用の public または protected フィールド: sample.Tax.test

このフィールドは決して使用されません。フィールドが public か protected なので、多分、それは解析の一部として見えないクラスで使用されることを意図しています。そうでなければ、クラスから除去することを検討してください。

```
package sample;

public class Tax {
    public String test;

    public static void main(String[] args) {
        int price = Integer.parseInt(args[0]);
        int consumptionTax = calcConsumptionTax(price);
        System.out.println(consumptionTax);
    }
}
```

# 終わったら

---

- ファイルに変更を加えてビルドを実行。
  1. 「5.プライベートリポジトリを作る」で作ったファルダをエクスプローラで開く。
  2. `sample-repository¥Tax¥src¥main¥sample¥Tax.java` または  
`sample-repository¥Tax¥src¥test¥sample¥TaxTest.java` をエディタで開く。
  3. 何かしら編集を加える。
    - 数字を変える
    - コメントを加える
    - メソッドを追加する。 etc...
  4. プッシュする。
  5. 再度 Jenkins でビルド実行を行う。



# 一旦3分休憩

---

- 後半は Jenkins にほぼ触れないです
- MagicPod を使うので MagicPod アカウントの確認をお願いします。
- 前回作成したアカウントでログイン

<https://magic-pod.com/accounts/login/>



The screenshot shows the Magic Pod login interface. At the top, there's a green header with the Magic Pod logo. The main content area is titled 'サインイン' (Sign In). Below the title, there's a message: 'GitHubアカウントを使用してサインインすることができます。または、Magic Podアカウントを作成し、サインインしてください。' (You can sign in using a GitHub account. Alternatively, create a Magic Pod account and sign in). There's a GitHub logo link. Below that, there are two input fields: 'ログイン:' (Login:) with a placeholder 'ユーザー名またはメールアドレス' (Username or email address), and 'パスワード:' (Password:) with a placeholder 'パスワード' (Password). There's a checkbox for 'ログインしたままにする:' (Keep me logged in:). At the bottom, there's a link 'パスワードをお忘れですか?' (Forgot your password?) and a 'サインイン' (Sign In) button. The footer contains copyright information, links to '利用規約' (Terms of Service), 'プライバシーポリシー' (Privacy Policy), 'ヘルプ' (Help), and 'チャットルーム' (Chat Room), along with the Trident Inc. logo.

# MagicPod について

---

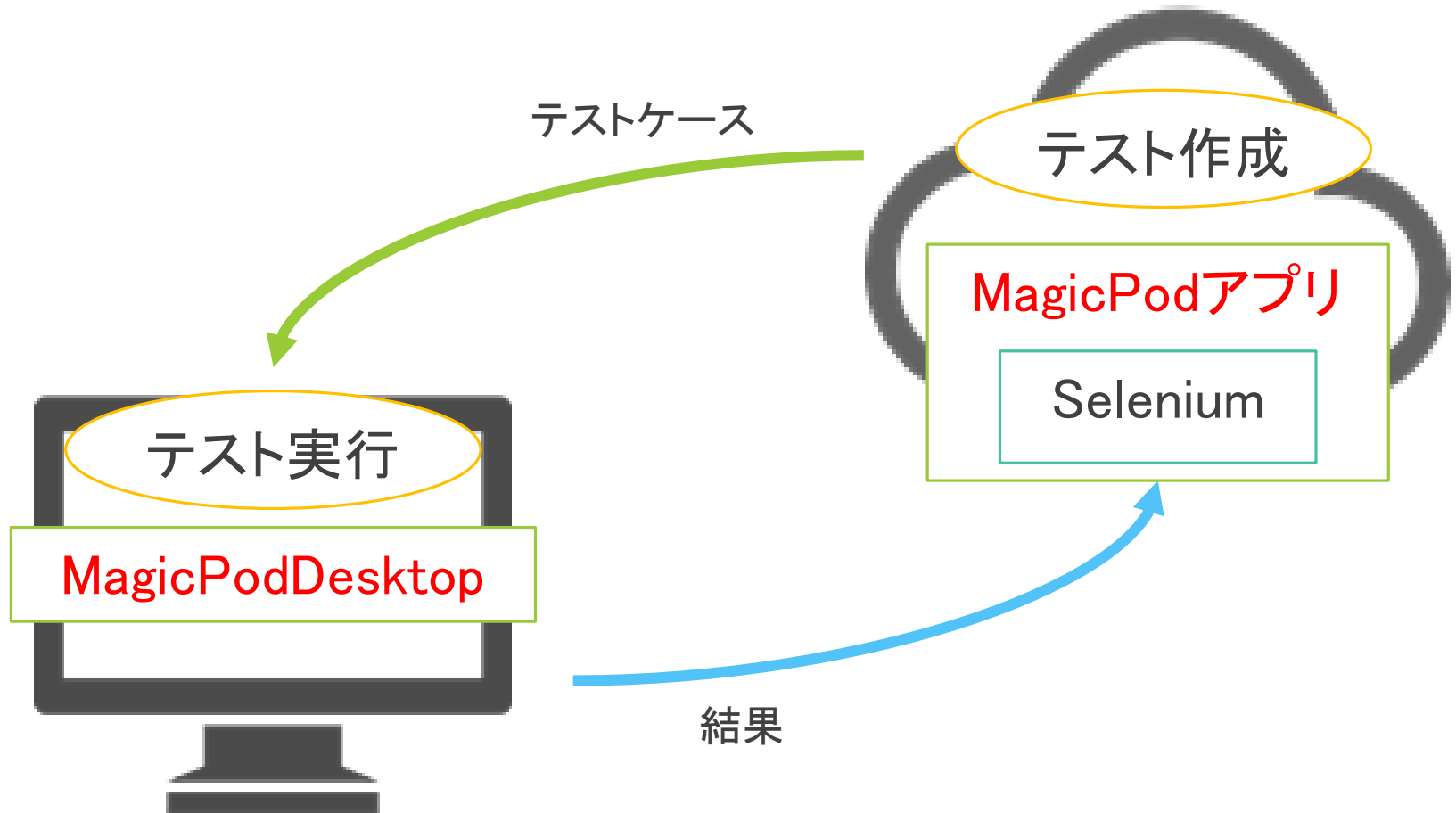
- Web画面の自動テストを行う
- 簡単操作でテストケースを作成
- ネットに接続できる環境であれば実行可能



Magic Pod

<https://www.magic-pod.com/> より引用

# MagicPodについて



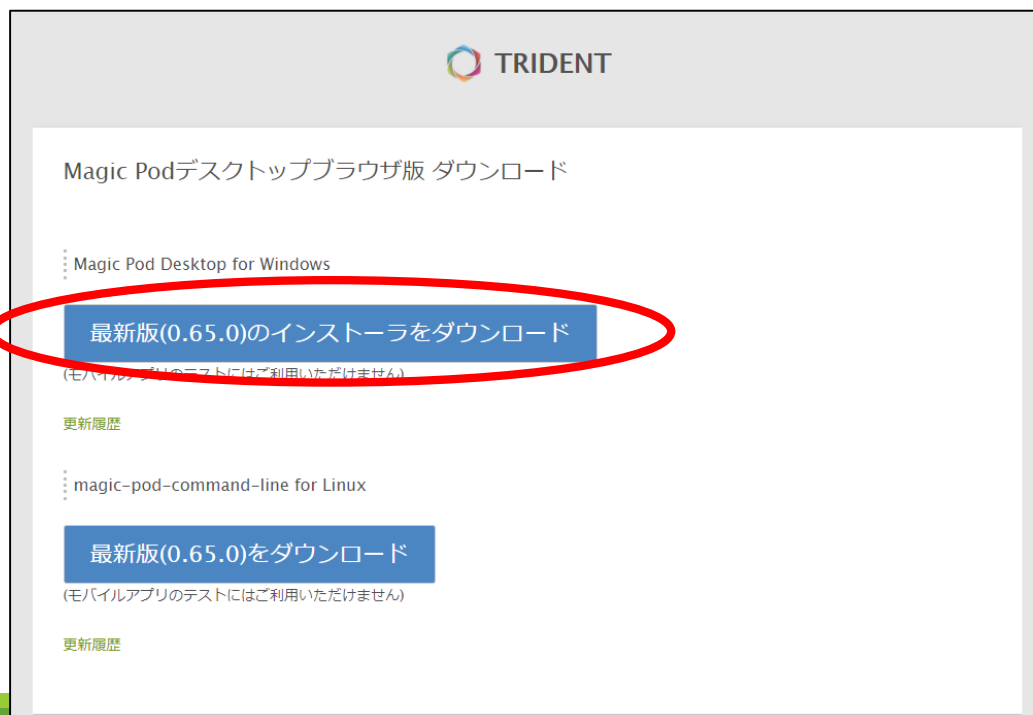
# MagicPod について

---

- 2月まで使用可能！
- 成果発表の際に使用することも可
- ホームページ  
<https://www.magic-pod.com/>
- 詳しい使い方  
<https://www.trident-qa.com/magic-pod-help/>

# MagicPodDesktop のダウンロード

1. <https://www.trident-qa.com/magic-pod-desktop-browser-downloads/> にアクセスする。
2. インストーラをダウンロードする。



# ここでデモンストレーション

---

ダウンロード中にデモを実施

# MagicPodDesktop のインストール

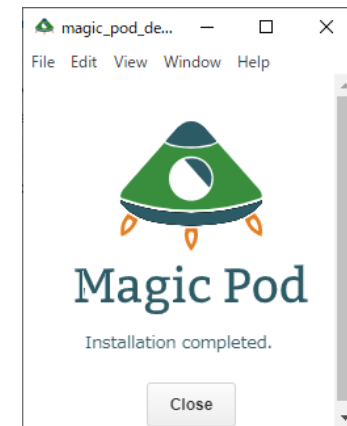
3. インストーラを実行する。



4. 「このデバイスに変更与えることを許可しますか？」と出るので、「はい」を押して許可する。

5. インストールが開始され、右のようなウィンドウが出れば完了！

6. 「Close」を押して閉じる。



# MagicPodに触れる

---

1. テストケースの作成
2. 自由に作る
3. 設定ファイルの確認



# テストケース作成

---

デモで紹介した内容を実際に作成！

# テストケースの作成

プロジェクトが作成されているので、そのプロジェクトに入る



The screenshot shows the Magic Pod web interface. The header is green with the Magic Pod logo and name. Below the header, there is a navigation bar with three tabs: "オーバービュー", "組織 1", and "プロジェクト 2". The "プロジェクト 2" tab is selected. The main content area is divided into two sections: "1 組織" and "2 プロジェクト". The "1 組織" section shows a list of organizations, with "JSP" being the only one visible. The "2 プロジェクト" section shows a list of projects, with "JSP / 先端技術研究会用テスト" being the only one visible. This project name is circled in red.

Magic Pod

オーバービュー 組織 1 プロジェクト 2

1 組織

JSP JSP

2 プロジェクト

JSP / 先端技術研究会用テスト

# テストケースの作成

自分のテストケースを作成するため、「追加」ボタンを押す

The screenshot shows the Magic Pod web interface. At the top is a green header with the Magic Pod logo. Below it, the JSP logo and the text 'JSP / 先端技術研究会用テスト' are displayed. A description reads '先端技術研究会で使用するプロジェクトです。'. A navigation bar contains tabs: 'テストケース 0', 'テスト一括実行 0', '共有ステップ 0', 'メンバー 1', and '設定'. The 'テストケース 0' tab is selected. On the right side of the page, a button labeled '追加' is circled in red. Below the navigation bar, a section titled '0個のテストケース' contains a message 'テストケースがありません'.

# テストケースの作成

JSP / 先端技術研究会用テスト

所有者: [REDACTED]

**作成**

① テストケース名:

説明:

② ラベル: ⚙️  
ラベルがありません

作成

← プロジェクトで表示されるテストケース名  
※ほかの人にも見えるので、  
自分のものだとわかるものにしてください。

# テストケースの作成

実際に作っていきましょう！



JSP / 先端技術研究会用テスト / テスト

オーバービュー    テスト結果 0    情報

作成者 [REDACTED]

テストケースを編集する

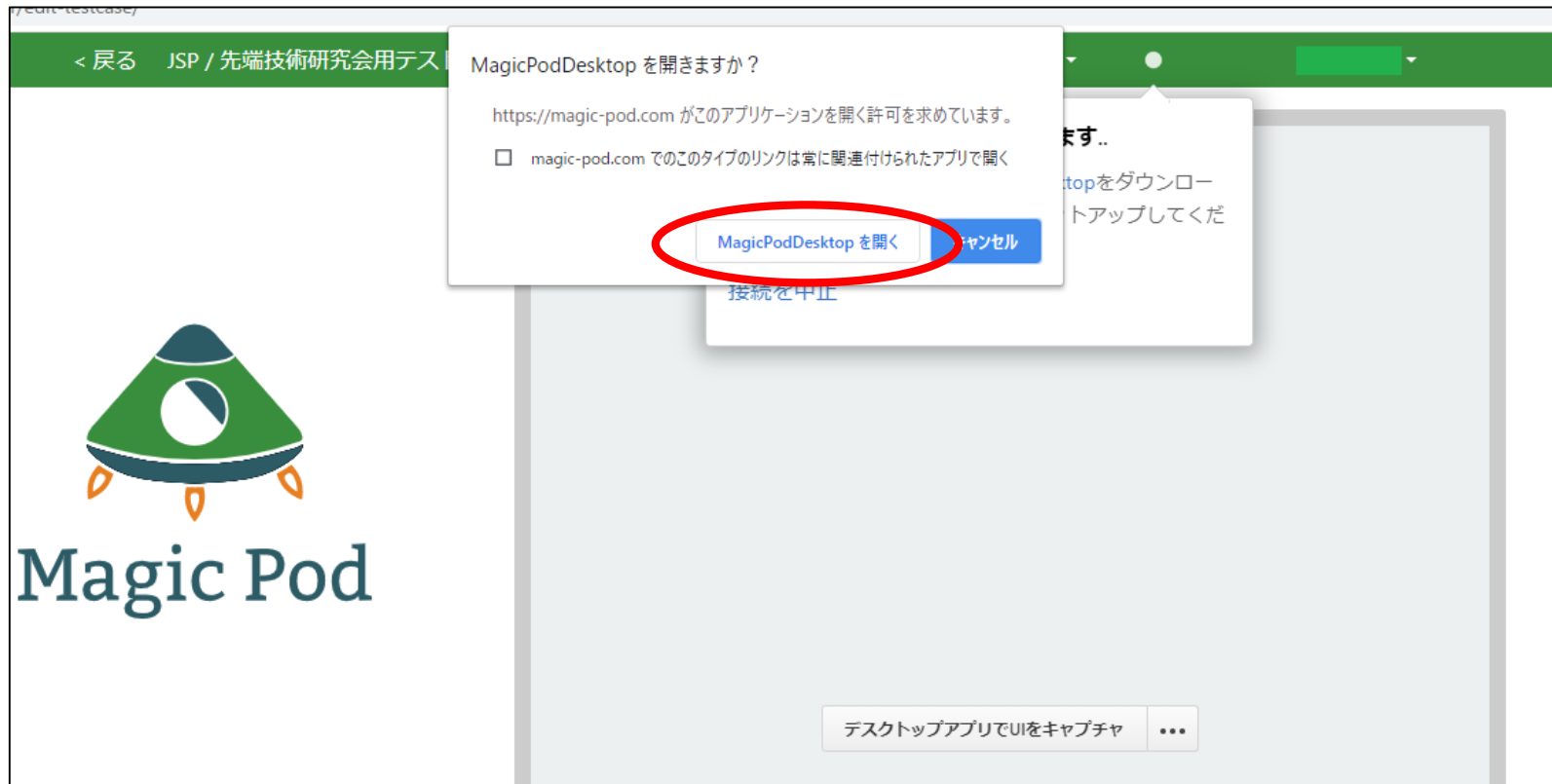
# テストケースの作成

まず MagicPodDesktop に接続する



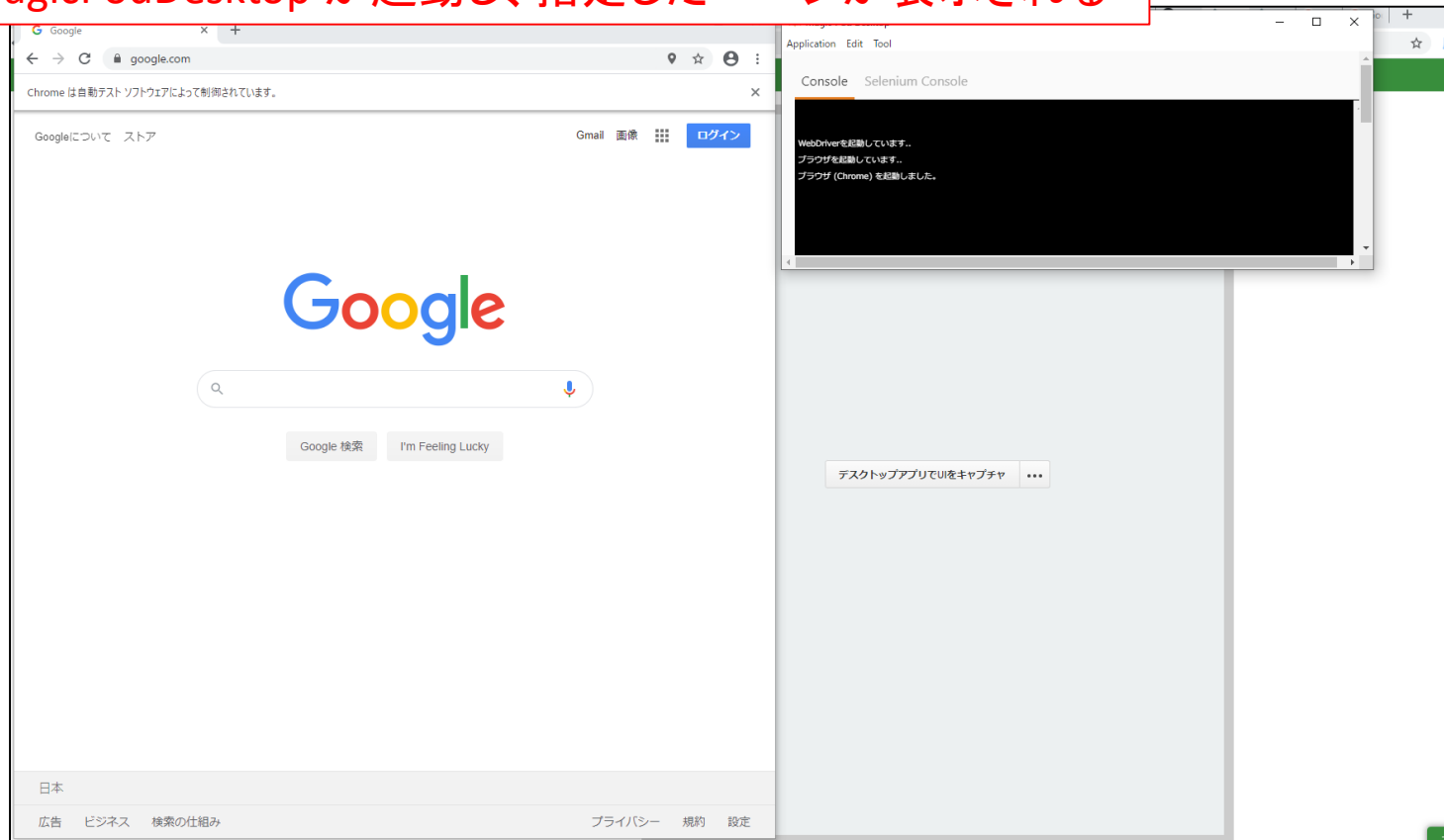
# テストケースの作成

## MagicPodDesktop を開く



# テストケースの作成

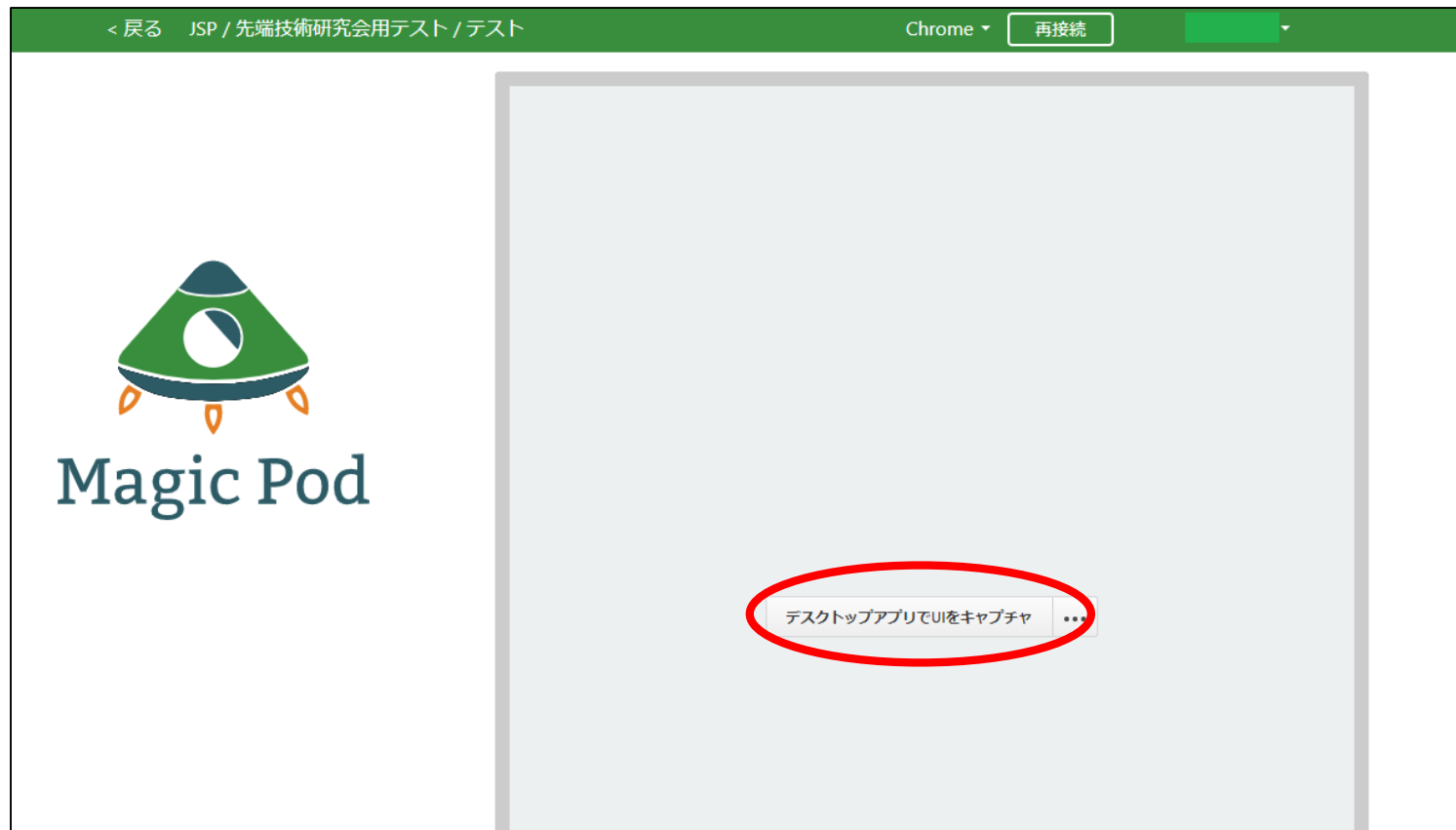
MagicPodDesktop が起動し、指定したページが表示される





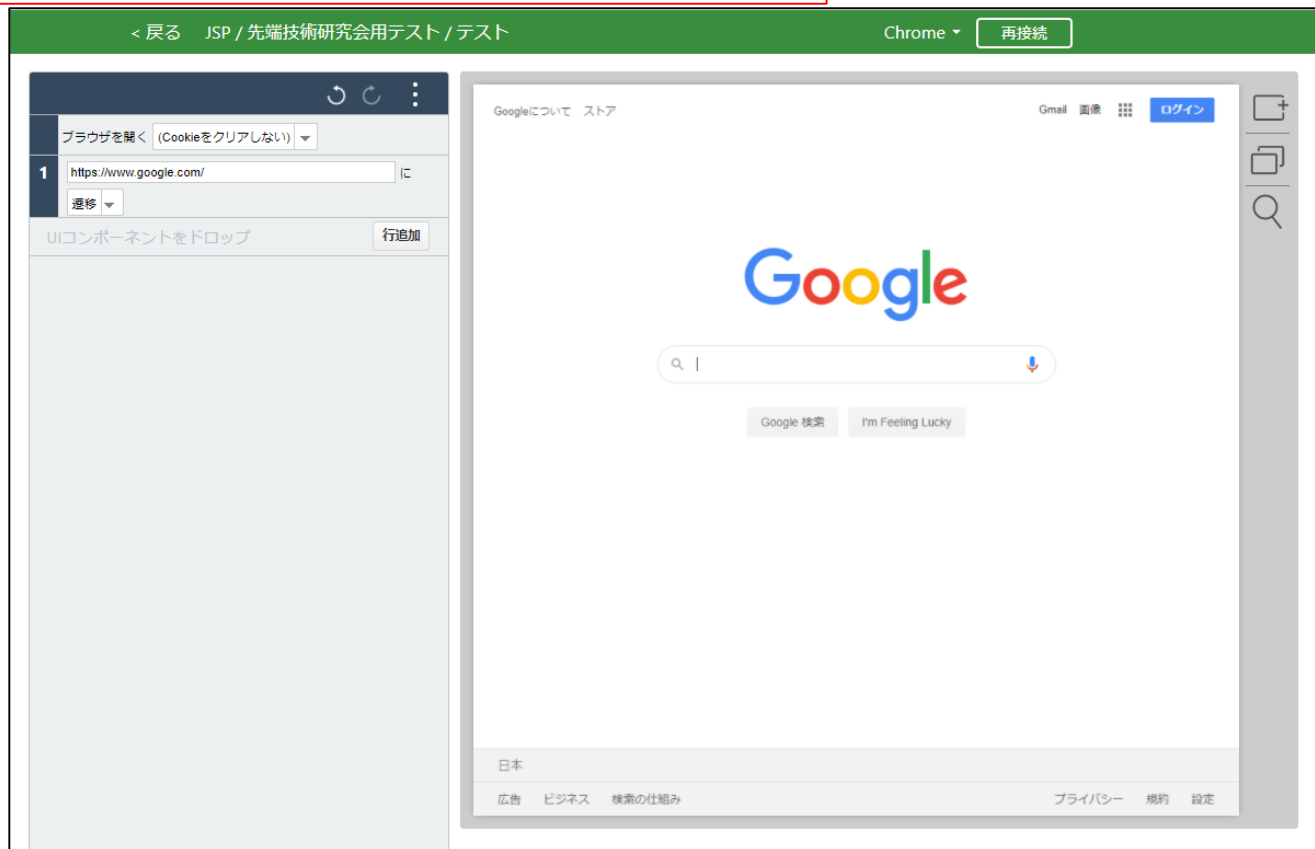
# テストケースの作成

戻って、画面のキャプチャを行う



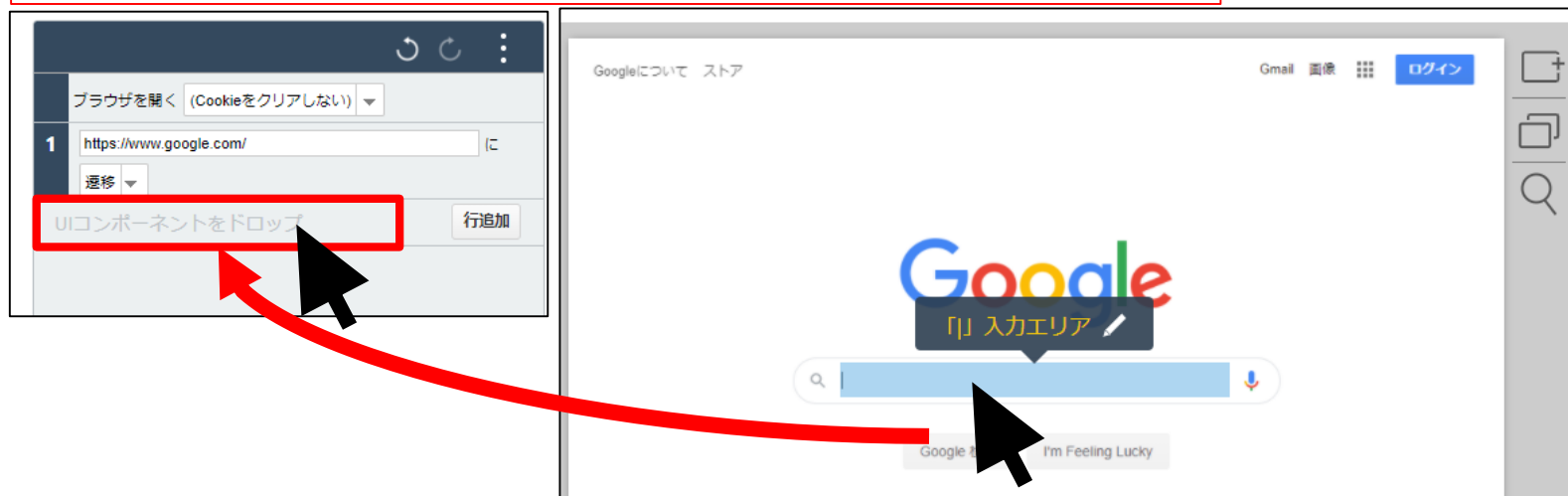
# テストケースの作成

これで準備完了！テストを追加しましょう。



# テストケースの作成

テキストボックスにカーソルを合わせ ドラッグアンドドロップ



このようになればOK

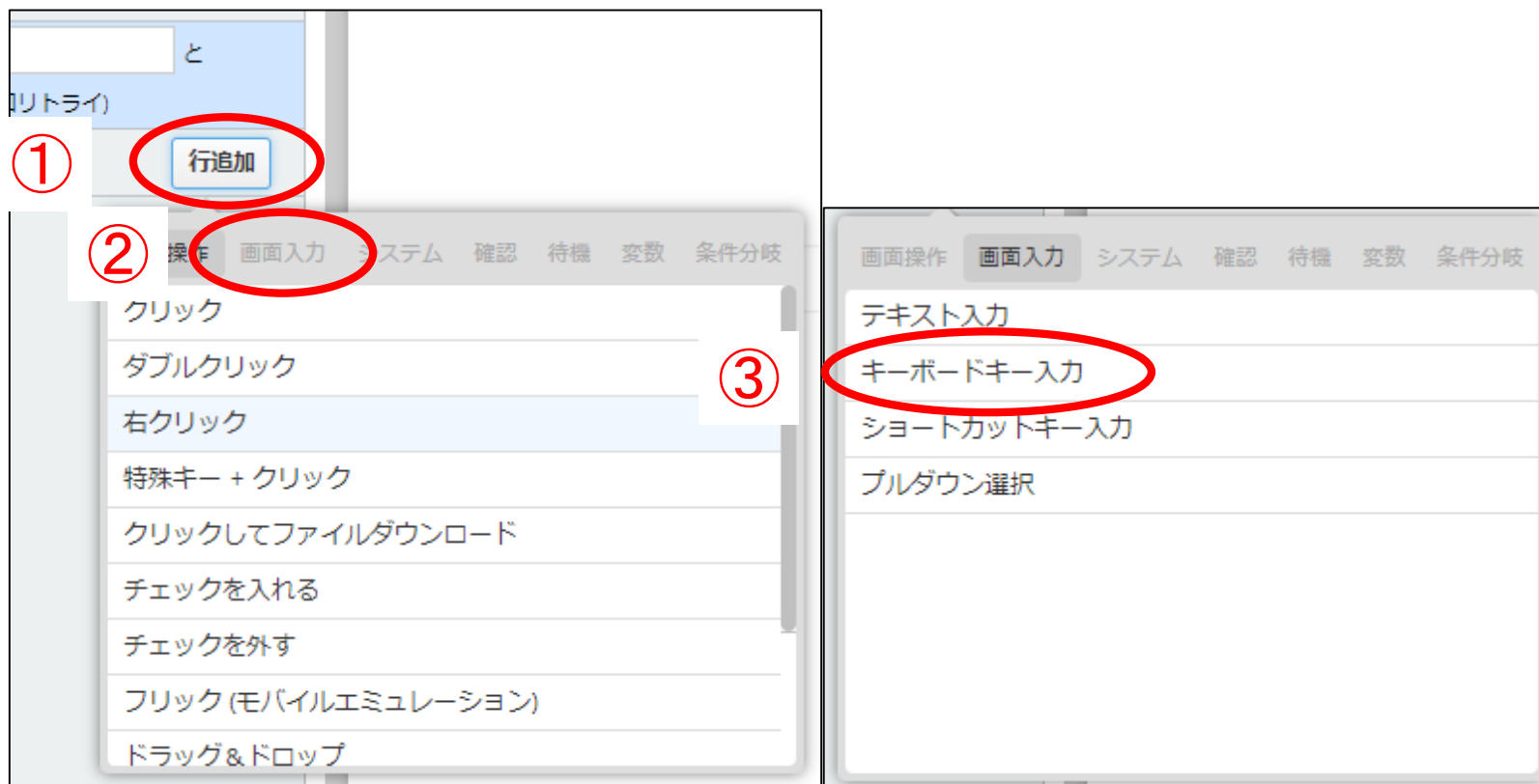
# テストケースの作成

検索するテキスト「株式会社ジェイエスピー」を入力する。



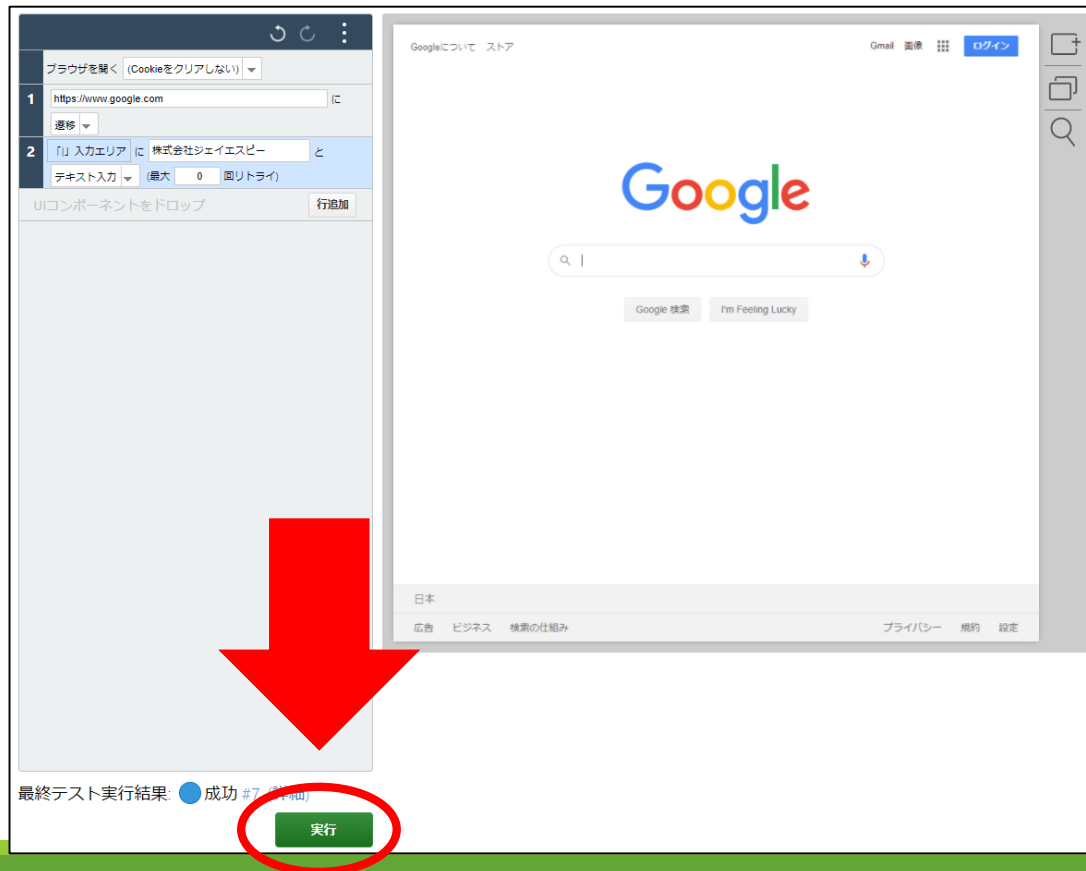
# テストケースの作成

エンターキーを押すケースを入れる。



# テストケースの作成

テストケースの作成が完了したので、実行する。



# テストケースの作成

Chromeが開き、テストが開始される



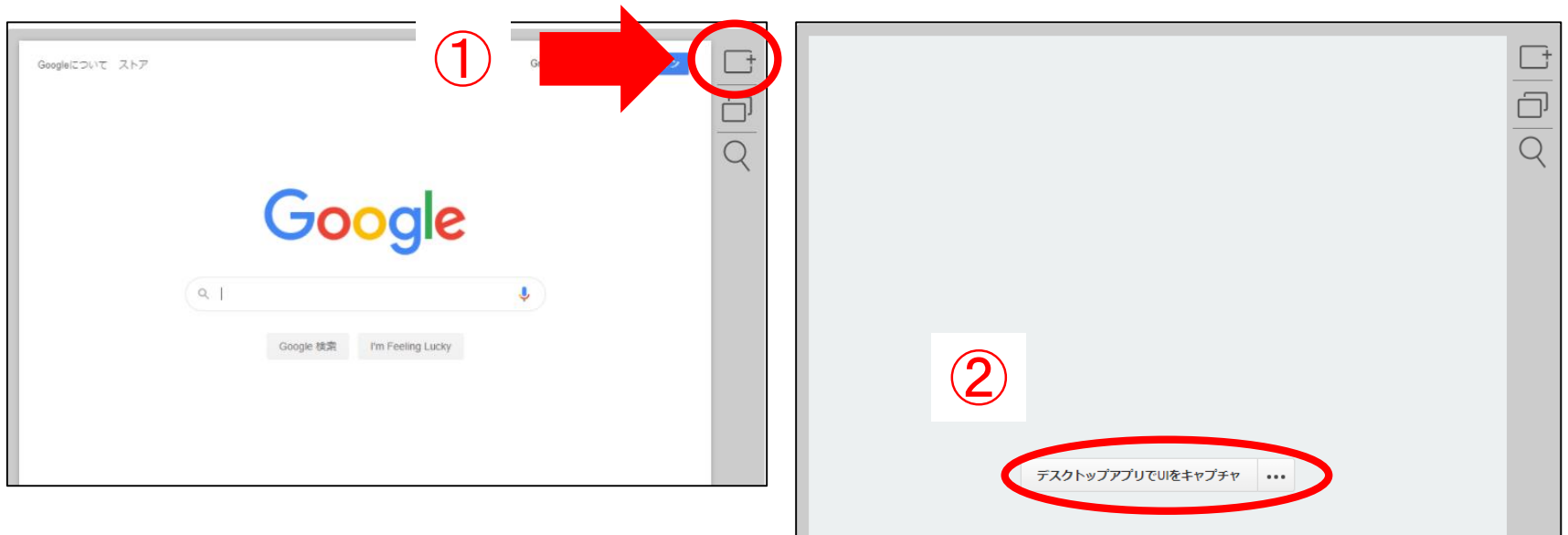
テスト終了後、MagicPodの画面で「成功」が出ていればOK

テスト実行結果: ● 成功 #3 (詳細)

成功

# テストケースの作成

次の画面をキャプチャする





# テストケースの作成

次の画面をキャプチャされる



# テストケースの作成



# テストケースの作成

再度実行してみる

The screenshot displays a web browser window with a Google search for '株式会社ジェイエスピー' (JSP Co., Ltd.). The search results show the company's website, 'www.jspnet.co.jp', and provide information about the company, including its establishment in 1980, its focus on IoT and sensor network technologies, and its headquarters in Tokyo. The left side of the image shows a test case execution interface with a list of steps: 1. ブラウザを開く (Cookieをクリアしない) (Open browser (do not clear cookies)), 2. 「」入力エリアに 株式会社ジェイエスピー と テキスト入力 (最大 0 回リトライ) (Enter 'JSP Co., Ltd.' in the input area (maximum 0 retries)), 3. エンター を キーボードキー入力 (Press Enter using keyboard input), and 4. 領域(101) を クリック (Click area (101)). Below the steps is a 'UIコンポーネントをドロップ' (Drop UI component) section and a '行追加' (Add row) button. At the bottom, it shows '最終テスト実行結果: ● 成功 #6 (詳細)' (Final test execution result: Success #6 (Details)) and a green '実行' (Execute) button.

ブラウザを開く (Cookieをクリアしない) ▼

1 https://www.google.com に 遷移 ▼

2 「」入力エリアに 株式会社ジェイエスピー と テキスト入力 (最大 0 回リトライ)

3 エンター を キーボードキー入力 ▼

4 領域(101) を クリック ▼

UIコンポーネントをドロップ 行追加

最終テスト実行結果: ● 成功 #6 (詳細)

実行

Google 株式会社ジェイエスピー

約 526,000 件 (0.38 秒)

www.jspnet.co.jp

横浜・東京のソフトウェア・システム開発 | 株式会社ジェイエスピー (JSP)

横浜・東京を中心にシステム開発を手掛けるジェイエスピーです。IoTセンシング技術を得意とし、センサーネットワークプラットフォーム、工場向けIoTシステム、在庫管理システム、スマートハウス基盤、見守りシステムなどの...

会社情報 ブログ 採用情報 アクセス

www.jspnet.co.jp > company

会社情報 | 株式会社ジェイエスピー (JSP)

社名 株式会社ジェイエスピー 設立 1980年 (昭和55年) 1月25日 代表者 稲田彰典 資本金 3,000万円 本社 〒220-0011 神奈川県横浜市西区高島2-6-32 横浜東口ウィズポートビル TEL: 045-444-3470; アクセスマップ 社員数 163名

資本業務提携先: 株式会社NTTデータイン... 社員数: 163名  
マージン率 (2017年度): 42.4% 資本金: 3,000万円

www.jspnet.co.jp > blog

ジェイエスピーのブログ | 株式会社ジェイエスピー (JSP)

株式会社ジェイエスピーが運営しているブログの一覧です。

www.jsp21.co.jp

株式会社ジェイエスピー 公式ホームページ

株式会社ジェイエスピーは愛知県豊田町で、小売販売、卸販売、通信販売、輸入、情報提供サービスなどポートビジネスを多角的に展開しています。

会社概要 事業 採用情報 社員の声

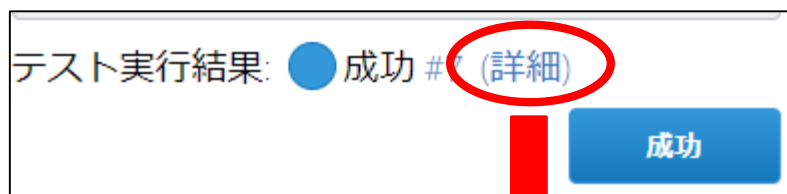
www.co-jsp.co.jp

株式会社JSP

サイトマップ 株式会社JSP 製品情報 企業情報 IR情報 グローバルネットワーク 採用情

# テストケースの作成

テスト終了後、MagicPodの画面で  
「成功」が出ていればOK



各ケースでのキャプチャは  
「詳細」から見れる



# 自由に作る

---

ほかにも様々な機能があります。

ぜひ触ってみてください。

「テストしたいページを変えたい」場合は次ページから紹介

# ページを変える

URLを変えるだけではなく、キャプチャも必要になる  
例として「 <https://www.jspnet.co.jp/> 」に変える



# ページを変える

The screenshot shows the MagicPodDesktop interface. On the left is a browser window with a green header bar containing a back arrow, the text "< 戻る JSP / 先端技術研究会用テスト / テスト", and a "Chrome" dropdown menu. A red circle highlights a "再接続" (Reconnect) button in the top right corner of the browser window. The browser's address bar shows "https://www.jspnet.co.jp/". Below the address bar is a list of actions numbered 1 to 4: 1. "ブラウザを開く (Cookieをクリアしない)" with a dropdown arrow; 2. "「」入力エリアに 犬 と" with a "テキスト入力" dropdown and "(最大 0 回リトライ)" text; 3. "エンター" with a dropdown arrow and "を キーボードキー入力" with a dropdown arrow; 4. "進む" with a dropdown arrow. Below the list is a "UIコンポーネントをドロップ" label and a "行追加" button. On the right side of the interface, there are two red-bordered boxes with red text. The top box contains "③ MagicPodDesktop に再接続する お目当てのページが開く". The bottom box contains "④ 再接続ができれば、ココを押す". Below the bottom box is a red circle highlighting a button labeled "デスクトップアプリでUIをキャプチャ".

< 戻る JSP / 先端技術研究会用テスト / テスト Chrome 再接続

1  に  
遷移

2 「」入力エリアに 犬 と  
テキスト入力 (最大 0 回リトライ)

3 エンター を キーボードキー入力

4 進む

UIコンポーネントをドロップ 行追加

③ MagicPodDesktop に再接続する  
お目当てのページが開く

④ 再接続ができれば、ココを押す

デスクトップアプリでUIをキャプチャ

# ページを変える

キャプチャ完了。  
次にそれぞれのケースでこのキャプチャを選択する必要がある

The screenshot shows a web browser interface with a URL bar and a list of actions. The '行追加' (Add Row) button is circled in red. Below the screenshot, two numbered instructions in red text are provided: ①「行追加」を押す and ②「画面操作」の「クリック」を押す.

①「行追加」を押す  
②「画面操作」の「クリック」を押す



# ページを変える

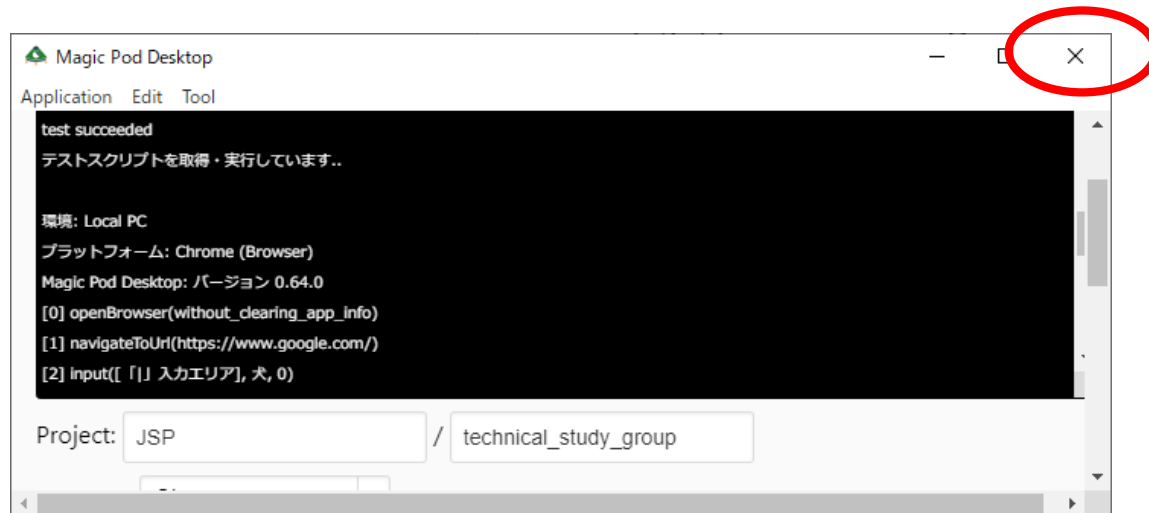


# 設定ファイルの確認

---

次回、JenkinsとMagicPodとの連携で使うファイルを確認します。

## ① MagicPodDesktop を閉じる

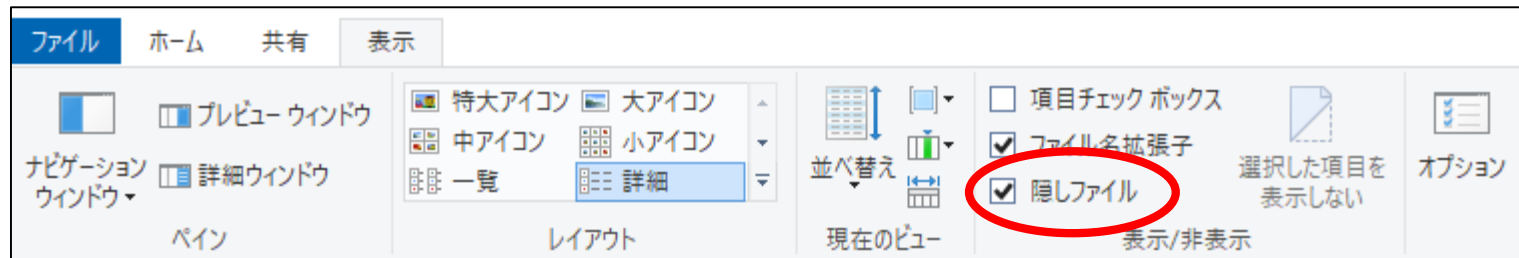


# 設定ファイルの確認

② 以下のディレクトリを開く。

C:¥Users¥(ユーザ名)¥AppData¥Roaming¥magic\_pod\_desktop

※隠しファイルの表示をONにする



# 設定ファイル

③ 「magic\_pod\_config.json」 を開く。

```
1  ↓
2  "owner": "JSP", ↓
3  ↓
4  ↓
5  ↓
6  "excludedTestCaseLabels": [], ↓
7  "sharedDataPatternRows": [], ↓
8  "capabilities": { ↓
9    "browserName": "chrome" ↓
10 }, ↓
11 "baseUrl": "https://www.jspnet.co.jp/", ↓
12 "envVars": [], ↓
13 "xmlTestOutput": true, ↓
14 "workDir": "C:¥¥Tools", ↓
15 "authTokenFilePath": "C:¥¥Users[redacted].magic_pod_token", ↓
16 "proxyServerUrl": "auto", ↓
17 "proxyServerAuthType": "none", ↓
18 "proxyServerAuthUser": "", ↓
19 "proxyServerAuthPassword": "", ↓
20 "logLevel": "beginner", ↓
21 "testCondition": { ↓
22   "device_type": "desktop", ↓
23   "browser": "chrome" ↓
24 }, ↓
25 "captureType": "on_each_step", ↓
26 "stepCaptureScope": "html", ↓
27 "sendMail": false, ↓
28 "retryCount": 0 ↓
29 } [EOF]
```

いろいろ書いてありますが、次回使います！

ここまで終わったら  
「手を挙げる」

# この研究会について

---

## 目標

「Jenkinsを使えるものにする手段」を知り、  
その利用価値を考える

## 成果物

Jenkinsで実現できることを提案  
→ グループで1つ。デモも OK

# 成果発表について

---

## 課題

### Jenkinsを使って困ったことを解決する方法を提案

- グループで資料やデモを作成
- 2月13日の第5回で発表(1グループ15分発表、質疑応答5分)
- 必ず Jenkins を使うこと(GitHubやMagicPodを使用してもよい)。
- ハードやツールの構成も発表に入れること
- 複数作成してもOK

# 成果発表について

---

## ■ 困ったことの内容は自由

- C# で開発して Jenkins で自動ビルドをしたい。
- 画面試験を Jenkins から自動で実施してほしい。
- 結果を Slack に通知してほしい。
- 日報を自動生成してほしい。 etc...

## ■ 様々なツール連携が可能。

## ■ こういうの実現したいということであれば伺います。

# ここでグループ決め

---

- 自分の番号を覚えてください。
- Jenkins の「パラメータ付きビルド」を使います。



# 次回以降について

---

	日程	内容	形式
第1回	10/10	CIについて講義／Jenkinsに触れる	オンライン（Zoom）
第2回	11/14	プラグイン／MagicPodの使用	オンライン（Zoom）
第3回	12/12	MagicPodとの連携／グループ話し合い	オンライン（Zoom）
第4回	1/16	Jenkinsのその他設定／グループ話し合い	オンライン（Zoom）
第5回	2/13	成果発表／懇親会	弊社会議室

- 時間： 10:00～12:00
- 詳細： 再度メールにてご連絡させていただきます。

# 皆様お疲れ様でした。

このミーティングは、13:00 まで開いております。  
質問等がございましたら、何なりとお申し付けください。

---