SAMIT18.05
GASによる初めてのボット作成

17043052 藤原 渓亮

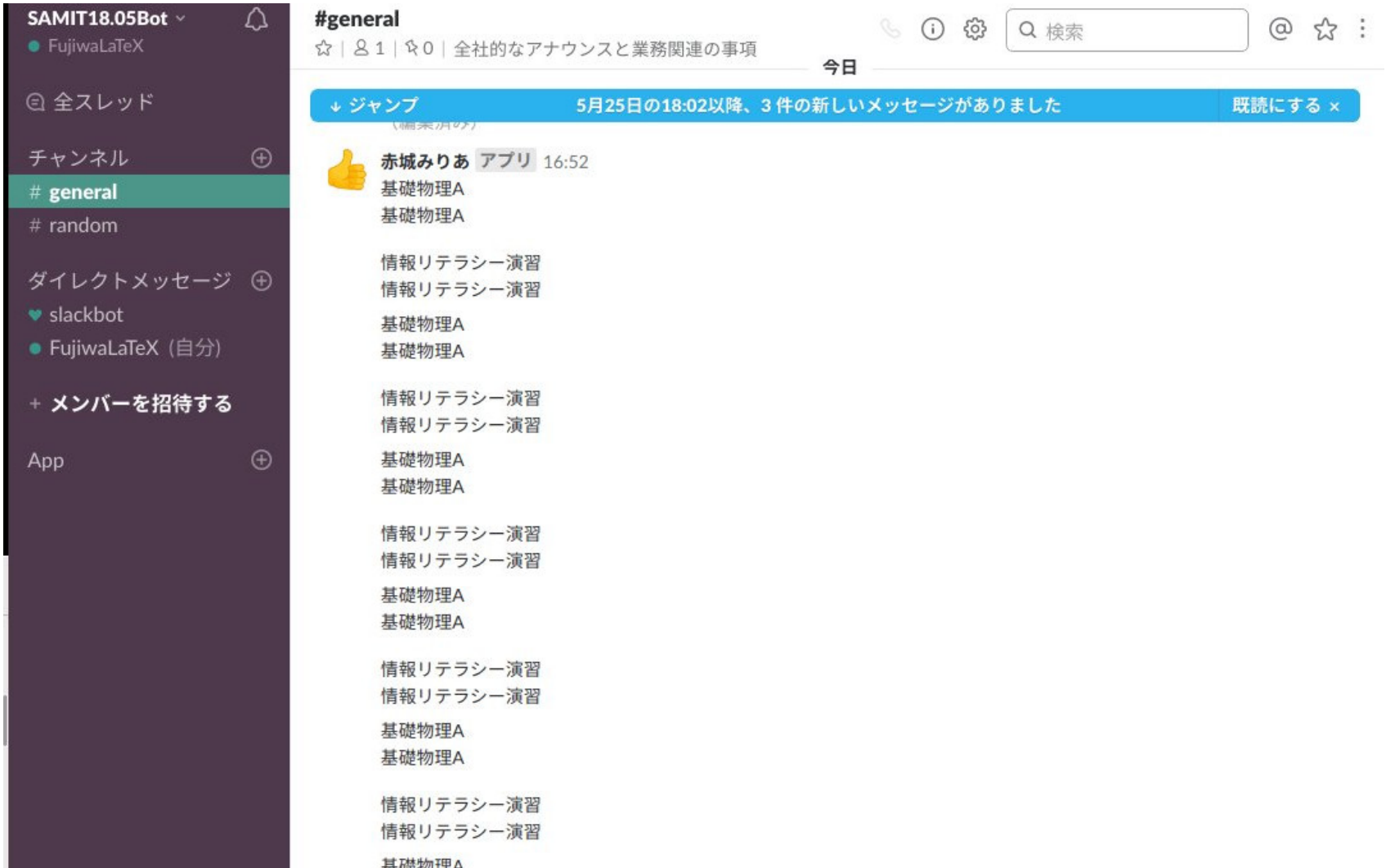
2018年5月26日

今日やること

ボット作ります

今日やること

こんな感じです.



今日やること

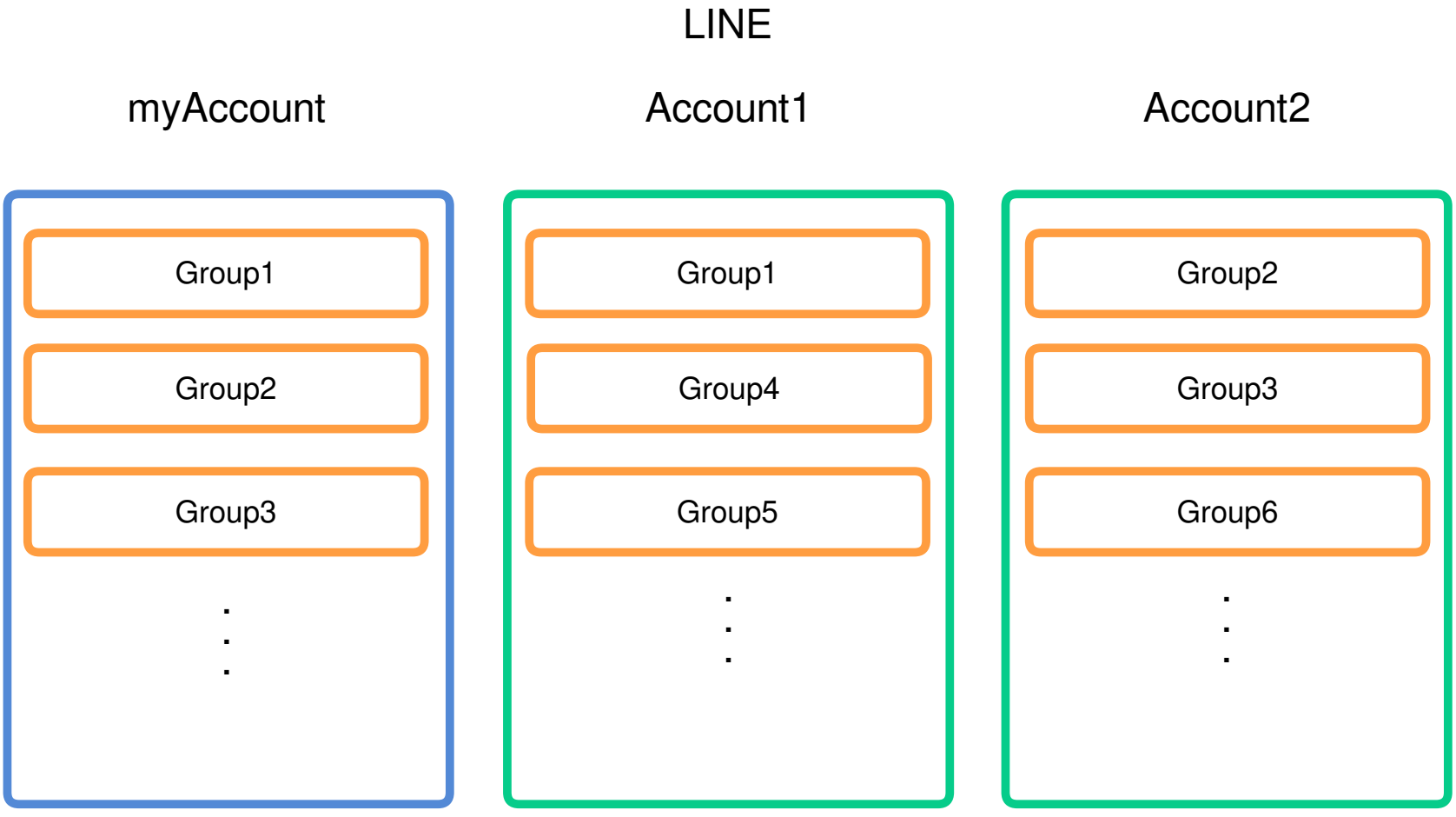
Slack(チャットツール) に定期的に時間割を通知してくれるボットを作ります.

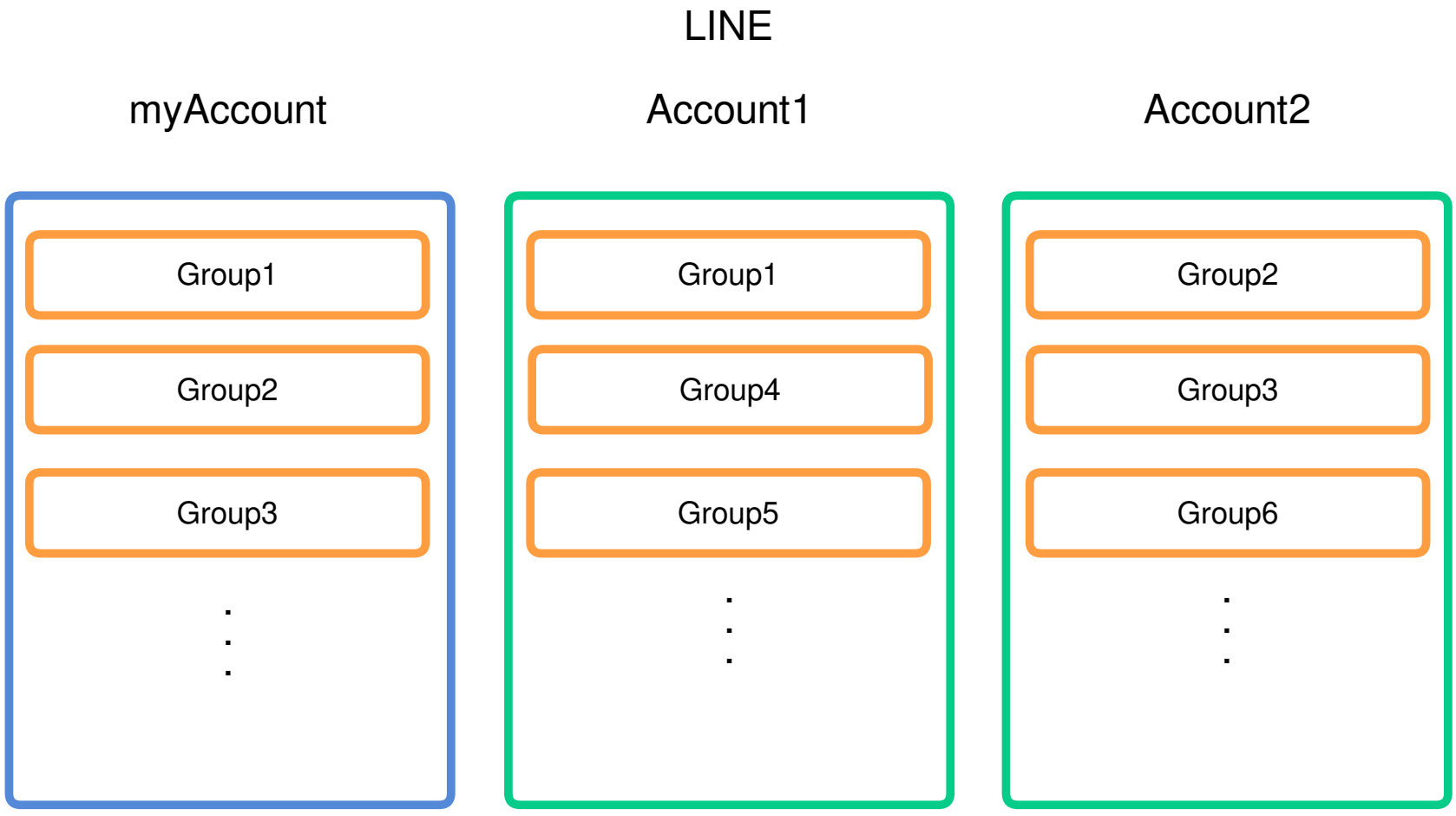
Slackの用意

Slackとは

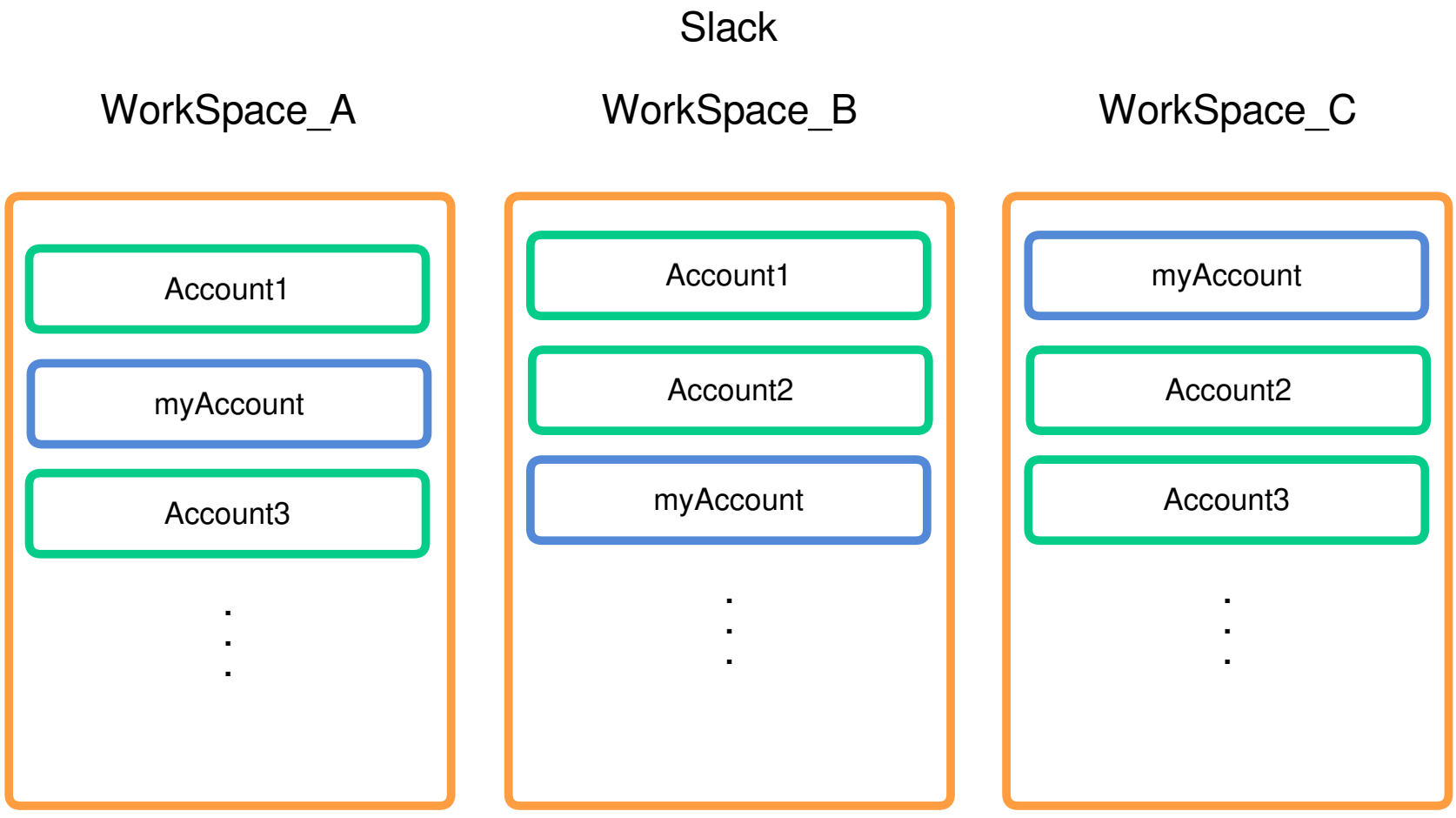
Slack Technologies社が開発したビジネス向けチャットツール

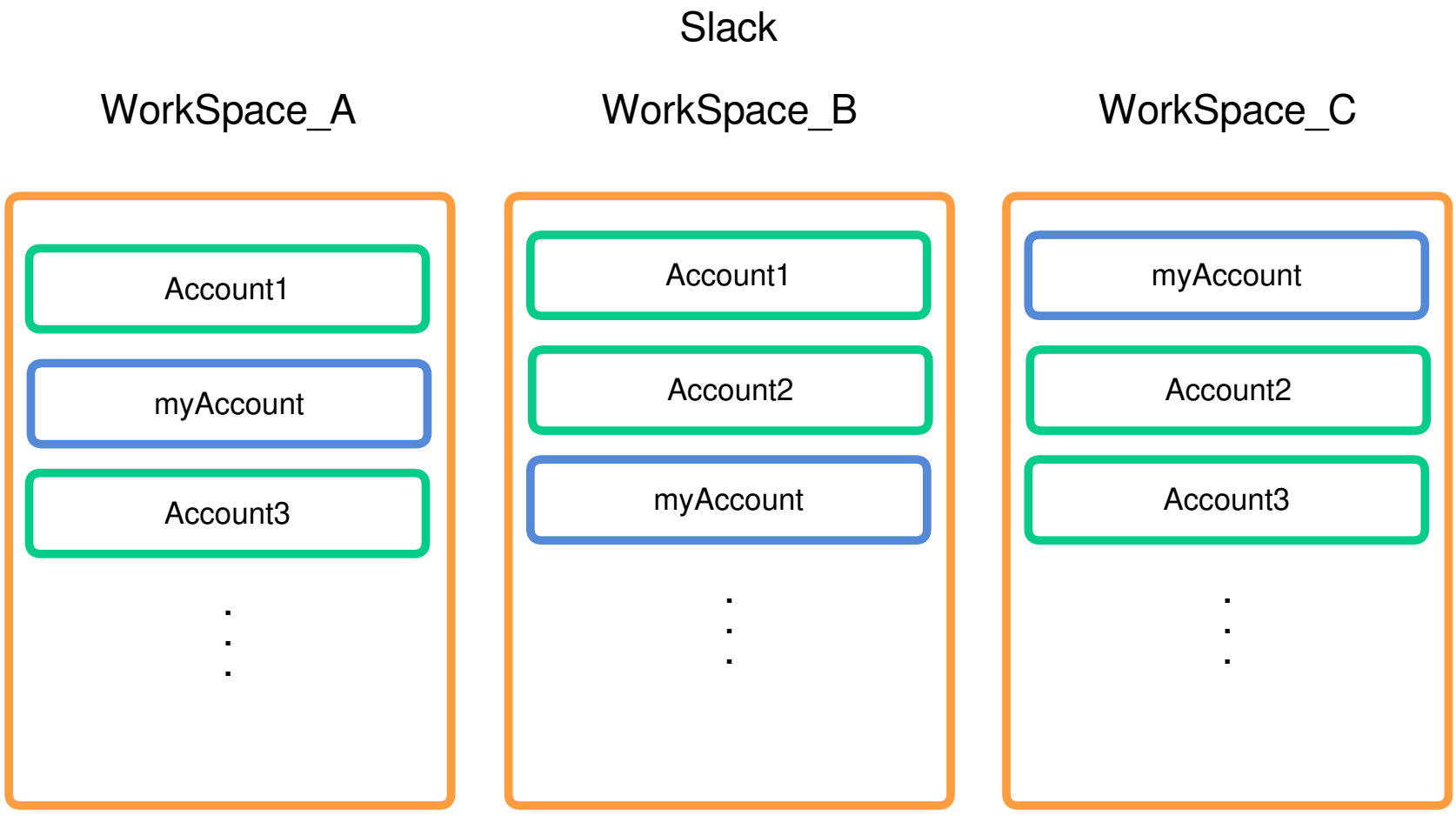
slack


LINE
myAccount
Account1
Account2
Group1
Group2
Group3
Group1
Group4
Group5
Group2
Group3
Group6



一つのアカウントが
複数のグループに所属


Slack
WorkSpace_A
Account1
myAccount
Account3
WorkSpace_B
Account1
Account2
myAccount
WorkSpace_C
myAccount
Account2
Account3



グループ毎にアカウントを持つ

ワークスペース作成

1. https://slack.com/intl/ja-jp にアクセス
2. 右上の「ワークスペース」をクリック
3. 最下部にある「ワークスペースの作成」をクリック

アプリ追加

1. チャット画面から「+アプリ追加」をクリック
2. “incomming web hook” で検索
3. 「設定を追加」をクリック

着信 Web フック

外部から Slack にメッセージを送れるアプリ

アプリ追加

1. チャット画面から「+アプリ追加」をクリック
2. “incomming web hook” で検索
3. 「設定を追加」をクリック

着信 Web フック

外部から Slack にメッセージを送れるアプリ

投稿するチャンネルを決めたら「インテグレーションの追加」

Slackにメッセージを送ってみる

コマンドラインからメッセージを送信

```
curl -X POST --data-urlencode
"payload={\"text\": \"Hello Slack\"}"
<your Webhook URL>
```

Slack にメッセージを送ってみる

コマンドラインからメッセージを送信

```
curl -X POST --data-urlencode
"payload={\"text\": \"Hello Slack\"}"
<your Webhook URL>
```

Windows は curl コマンドありません.

Googleスプレッドシートをオリジナル時間割にする

スプレッドシートをコピー

1. https://goo.gl/ec1Gmk からスプレッドシートにアクセス
2. 「ファイル」 → 「コピーを作成」
3. 自分のディレクトリに保存
4. コピー先で時間割カスタマイズ

エクセルと使い方はほぼ一緒

GAS について

GAS とは

Google Apps Script の略で，Google のサービスを操作できる環境．言語は JavaScript

GAS から利用できるサービス

Google スプレッドシート，Google ドキュメント，Google スライドのデータ操作や取得，Gmail へ通知を送信などが可能

スプレッドシートにGASを導入

1. コピーしたスプレッドシートを開く
2. 「ツール」 $\rightarrow$ 「スクリプトエディタ」

GASを書く

Hello World!!

```
1 function helloWorld(){
2   Logger.log("Hello World!!");
3 }
```

スクリプト実行

1. 「実行」→「関数を実行」→ “helloworld”
2. ctrl + Enter でログ確認

GASを書く

スプレッドシートからデータ取得

```
1  function helloWorld(){
2    Logger.log("Hello World!!");
3  }
4  function getSheetInfos(){
5    const values = SpreadsheetApp.getActiveSheet().getRange("A1").getValues();
6    return values;
7  }
```

A1セルから値を取得して，ログに出力
(ctrl+Enterで見れる)

GAS から Slack へメッセージを送る

GAS から POST リクエストを送る

```
1  function postSlack(text){
2    const url = "<your Webhook URL>";
3    const options = {
4      "method" : "POST",
5      "headers": {"Content-type": "application/json"},
6      "payload" : JSON.stringify({"text": text})
7    };
8    UrlFetchApp.fetch(url, options);
9  }
```

GAS から Slack へメッセージを送る

GAS から POST リクエストを送る

```
1  function postSlack(text){
2    const url = "<your Webhook URL>";
3    const options = {
4      "method" : "POST",
5      "headers": {"Content-type": "application/json"},
6      "payload" : JSON.stringify({"text": text})
7    };
8    UrlFetchApp.fetch(url, options);
9  }
```

送るメッセージを受け取る関数

```
1  function test(){
2    postSlack("これはテストです");
3  }
```

GAS から Slack へメッセージを送る

GAS から POST リクエストを送る

```
1  function postSlack(text){
2    const url = "<your Webhook URL>";
3    const options = {
4      "method" : "POST",
5      "headers": {"Content-type": "application/json"},
6      "payload" : JSON.stringify({"text": text})
7    };
8    UrlFetchApp.fetch(url, options);
9  }
```

UrlFetchApp.fetch

url で指定した場所に options で指定した条件でデータを送る

GAS から Slack へメッセージを送る

GAS から POST リクエストを送る

```
1  function postSlack(text){
2    const url = "<your Webhook URL>";
3    const options = {
4      "method" : "POST",
5      "headers": {"Content-type": "application/json"},
6      "payload" : JSON.stringify({"text": text})
7    };
8    UrlFetchApp.fetch(url, options);
9  }
```

url

データを送る先

GAS から Slack へメッセージを送る

GAS から POST リクエストを送る

```
1  function postSlack(text){
2    const url = "<your Webhook URL>";
3    const options = {
4      "method" : "POST",
5      "headers": {"Content-type": "application/json"},
6      "payload" : JSON.stringify({"text": text})
7    };
8    UrlFetchApp.fetch(url, options);
9  }
```

options

- method HTTP リクエストの種類
- headers 送るデータの形式
- payload 送るデータの内容

GAS から Slack へメッセージを送る

GAS から POST リクエストを送る

```
1  function postSlack(text){
2    const url = "<your Webhook URL>";
3    const options = {
4      "method" : "POST",
5      "headers": {"Content-type": "application/json"},
6      "payload" : JSON.stringify({"text": text})
7    };
8    UrlFetchApp.fetch(url, options);
9  }
```

JSON

データ記述言語の一種 (データ構造を表す)

月曜日の時間割を Slack に通知してみる

月曜日の時間割 → B2 から B11

月曜日の時間割を返す関数

```
1  function notifyMondaySchdule(){
2    const classes = getSheetInfos("B2:B11");
3    const todayScheduleStr = listToStr(classes);
4    postSlack(todayScheduleStr);
5  }
```

リストを文字列にする関数

```
1  function listToStr(strList){
2    const strs = strList.reduce(
3      function(previousValue, currentValue){
4      return previousValue + "\n" + currentValue;
5    });
6    return strs;
7  }
```

今日の時間割を Slack に通知してみる

今日の時間割を返す関数

```
function notifyTodaySchedule(){
  const weekDayList = {
    '日':'B','月':'B','火':'C','水':'D','木':'E','金':'F','土':'B'
  };
  const today = getDay();
  const rowNumber = weekDayList[today];
  const classes = getSheetInfos(rowNumber+"2:"+rowNumber+"11");
  const todayScheduleStr = listToStr(classes);
  postSlack(todayScheduleStr);
}
```

今日の曜日を返す関数

```
function getDay(){
  const weekDayList = ["日", "月", "火", "水", "木", "金", "土"];
  const dateObj = new Date() ;
  const weekDay = weekDayList[ dateObj.getDay() ];
  return weekDay;
}
```

定期的に時間割を Slack に通知してみる

1. 時計のアイコンをクリック → 「トリガーが設定 . . .」をクリック
2. 実行 → “notifyTodaySchedule” へ
3. イベント → 「時間主導型」,「分タイマー」,「1 分ごと」
4. 保存

約 1 分待つ

携帯に通知を送る

1. Android/iPhone に “Slack” をインストール
2. “Slack” を開いて「ワークスペースを追加」
3. 今回作成したワークスペースに登録したアドレスを入力
4. 「参加しているワークスペース」からワークスペース選択