



MLH Localhost – Build Apps for Slack:

Presenter's Guide

This is your troubleshooting guide for the **MLH Localhost – Build Apps for Slack**. We will create a Two Truths and a Lie app that you can play with other people in your workplace. This guide is here to help you with any issues you may encounter in the code or the process of setting up your app.

SECTION 1: CREATING A SLACK ACCOUNT

- To create a Slack account you need to enter your email address and then confirm your account. If the verification email does not show up, check your sp.

SECTION 2: CREATING A GLITCH ACCOUNT

- You will need to create a Glitch account to complete this exercise. This must be done with either a Facebook or Github account. To proceed, make sure you have one of these accounts.

SECTION 3: CREATING AN APP

- Do you not see your app? Double check to make sure you are in the right workspace.

SECTION 4: ADD A BOT USER

- Do you not see your bot? On the Top Left corner, make sure that the “TwoTruthsandaLie” app is selected as the app you are working on in the Bot User section.
- Keep the standard selection of “off” when asked whether you want to display your Slack bot as always online.

SECTION 5: INSTALL THE APP TO YOUR WORKSPACE

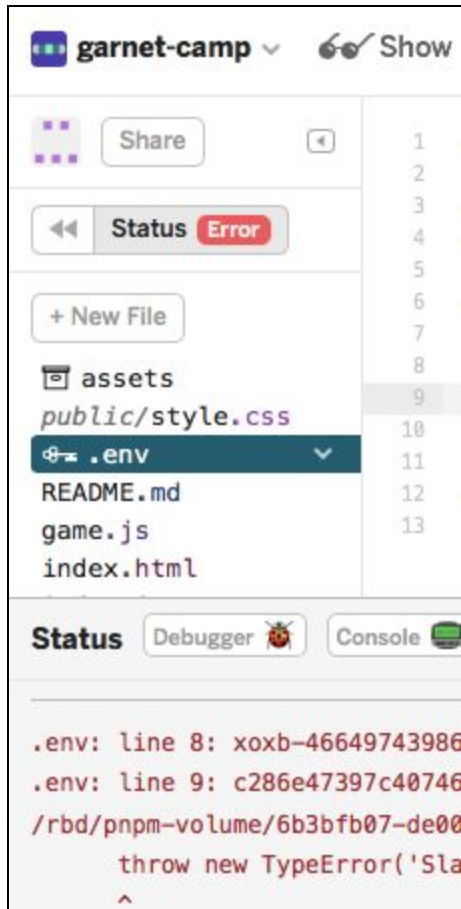
- If you have trouble with the installation, check which authorization you are using. Make sure that you are using specifically the Bot OAuth instead of the general OAuth token.

SECTION 6: INSTALL THE APP ON YOUR WORKSPACE

- You need to put two different keys into the .env space where the Token and Signing Secret are indicated. These keys are in two different places. The Slack Token is located in the OAuth permissions tab under “Bot User OAuth token.” Meanwhile, the “Signing Secret” is located in the app admin panel.

- If you get an Error check whether you have copied the entire key or whether you have deleted the placeholder text.

SECTION 7: VERIFYING YOUR IDENTITY TO THE SLACK API



- If you are unable to paste your copied Token, make sure that you have selected “Remix This Code” on the upper Right Hand Side of Glitch.
- If you have an error message when you paste your Verification Token and Signing Secret, double check exactly how you entered your Signing Secret and Slack Token. Make sure there is no space between the equal sign and the beginning of the Slack Token and Signing Secret. An extra space will create a Status error.

```

1  # Environment Config
2
3  # store your secrets and
4  # only invited collabora
5
6  # reference these in you
7
8  SLACK_TOKEN=xoxb-46649743
9  SLACK_SIGNING_SECRET=c286

```

- You should get “The verification server is now listening at the URL: <http://:::3000/events>” displayed in your console when you have copied `“./node_modules/.bin/slack-verify --secret=$Signing_Secret --port=3000 --path=/events”` correctly and pasted it into your terminal. What is happening is that our computer is making a call to Slack which contains a verified code (think like a secret handshake) that lets Slack know that your computer should be trusted. Remember to post this into Glitch’s Console, not your own terminal.
- If you do not get a message that the server is working, check to make sure that you have deleted the “\$” symbol and that there are no spaces between the Signing Secret key and the equals sign.

Your console should appear as follows:

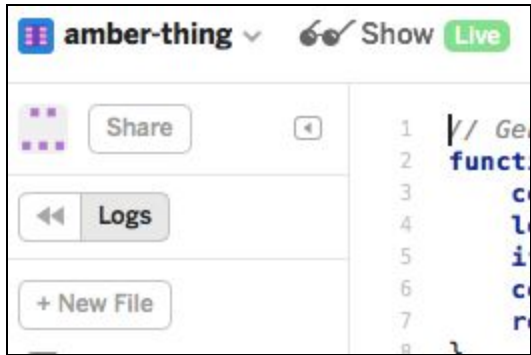
```

$ ./node_modules/.bin/slack-verify --secret=c286e47397c40740551391900d219c10
The verification server is now listening at the URL: http://:::3000/slack/events

```

SECTION 8: VERIFYING YOUR IDENTITY TO THE SLACK API


- To verify your identity to the Slack API you need to access the Console on Glitch. You should be able to do this by selecting “Show Live.” If you have done this in previous steps, you will need to select the Logs button on the Left panel and then Console.



- In the Console, you will need to enter the command:
`./node_modules/.bin/slack-verify --secret=$SLACK_SIGNING_SECRET
--port=3000 --path=/events`

In order to reduce the chance of bugs, copy this address into Notepad, add in your Slack Signing Secret where the URL says `$SLACK_SIGNING_SECRET` and then copy the altered url into your console.

- As you verify your app, you need to take the URL from the website that is displayed when you click **Show Live** on Glitch and then add `/events` to the end of the URL. If you do not add `/events`, you will get a 500 error that the challenge parameter is not responding. If you get this error, double check your url.

 Your request URL gave us a 500 error. Update your URL to receive a new request and challenge value.

Enable Events

On

Your app can subscribe to be notified of events in Slack (for example, when a user adds a reaction or creates a file) at a URL you choose. [Learn more.](#)

Request URL Your URL didn't respond with the value of the `challenge` parameter.

Retry

We'll send HTTP POST requests to this URL when events occur. As soon as you enter a URL, we'll send a request with a `challenge` parameter, and your endpoint must respond with the challenge value. [Learn more.](#)

SECTION 9: SELECT WORKSPACE EVENTS

- In later stages a part of the “TwoTruthsandaLie” app might not work if you do not have the proper Event names selected. Double check that you selected `member_joined_channel`, `message.channels`, and `message.groups`.
- Double check that you have Selected **Save Changes**. When you do, there will be a drop down banner that will remind you that you need to reinstall your app before changes take effect. We will reinstall the app in later steps.

You've changed the permission scopes your app uses. Please [click here](#) to reinstall your app for these changes to take effect (and if your app is listed in the Slack App Directory, you'll need to resubmit it as well).

SECTION 10: INTERACTIVE COMPONENTS

- Again, you will need to enter the url from your live glitch website, and add `/actions` at the end of the URL. Your screen should appear like this:

Interactive Components

Interactivity

On

Any interactions with actions, dialogs, message buttons, or message menus will be sent to a URL you specify. [Learn more.](#)

Request URL

`https://garnet-camp.glitch.me/actions`

We'll send an HTTP POST request with information to this URL when users interact with a component (like a button or dialog).

- Unlike with Events, the web page will not display that you have an error until your app does not work in later steps, so double check now that you are using the correct URL.

SECTION 11: ADD SCOPES TO YOUR APP

- It might be confusing that when you add Scope permissions in your app, the Scope appears on the Right side but is also listed as **Conversations** and **Interactivity**.

Select Permission Scopes

bot

Conversations

Send messages as TwoTruthsandaLie chat:write:bot

Interactivity

Add a bot user with the username @twotruthsandalie (already added) bot

Users

Access your workspace's profile information users:read

- After you **Save Changes**, make sure to reinstall the app or else changes made in the last three steps will not appear. If you navigate away from the web page without Authorizing your app again, you will need to add another permission or delete and re-add a permission to Re-Authorize.

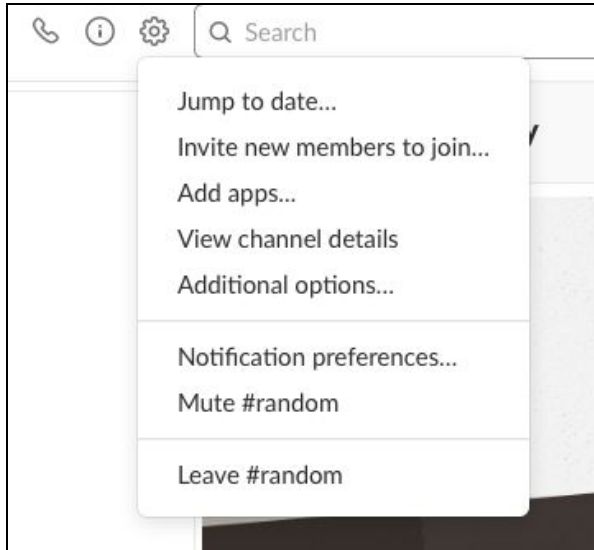
SECTION 12: APP TRIGGERING

- When you try your app for the first time in your workspace, nothing will happen. The MLH team put some bugs intentionally in the code so you can get some experience understanding how to find and correct bugs. To get the app to trigger, we will need to change some lines in `game.js`. Scroll down to the section of `"this.web.dialog.open"` and change `"blannk_trigger_id", dialog` to `payload.trigger_id, dialog`. The code should appear like below. What we are doing is connecting the uam with ser action to a function in the code that would display the dialog that will start the game.

```
this.web.dialog.open({trigger_id: payload.trigger_id, dialog});
const msg = payload.original_message;
msg.attachments[0].text = 'Awesome!';
msg.attachments[0].actions = [];
respond(msg);
```

- When you have changed this code, your app will still not work. You will need to both Reinstall App and delete and reinstall the **#twotruths** channel.

To Reinstall App, go to the Slack API page. Under **OAuth & Permissions** Select **Reinstall App**. You will also need to delete the **#twotruths** channel and add it again for the app's changes to take effect. You delete the **#twotruths** channel by going to Settings at the Top Right corner of the channel and going to **Leave Channel**.



SECTION 13: EPHEMERAL CREATION

- To see the changes made to your app, you will need to repeat the above process to reinstall your app through the Slack API, delete your channel and add your channel back to your Slack dashboard.
- If your app is still not working, quit the server, re-save your **SLACK_TOKEN** and **SLACK_SIGNING_SECRET** variables, and start the server again.

BACKGROUND

Slack was created in August of 2013 by Stewart Butterfield, Eric Costello, Cal Henderson, and Serguei Mourachov and stands for “Searchable Log of All Conversation and Knowledge.” Slack began as a way for the team of Glitch (then a massive multi-player online game) to communicate between each other. Even as the team realized that Glitch was an Adobe flash based game at a time the market had moved towards mobile app-based games, they found that the internal software they developed could be used to help teams collaborate.

Slack’s architecture contains three different kinds of interactions between users. There are **direct messages** between individuals. There are **channels** which are like big group messages that are centered around one central idea that the conversation is tagged with. Example tags include #general, #random, and #marketing. Finally there are threads which are like public versions of direct group messages that are also searchable. This creates a way for people to identify a topic of interest in a thread, have a conversation with a smaller group of people in a thread, and then bring the thread back into the mainstream of conversation when a decision has been made.

Standing outside of these different kinds of messaging types are **apps** which can be integrated into Slack. The Google drive app allows Slack users to import documents, Trello to help Slack users track of projects, and Giphy, to import fun GIFs into conversations. In this workshop participants are developing an app which can be used in either direct messages, channels, or threads.

In this workshop a central idea is how to use Glitch and then how to install the app created on Glitch to a Slack workspace. Glitch is a platform sort of like Youtube, where individual collaborators can upload their work and others can search through others’ work. A key difference is that Glitch is for apps and websites, and on Glitch the apps that are shared are meant to be changed by anyone, or “remixed.”

DEFINITIONS

Below are several concepts defined that are used in the workshop.

- API (Application Programming Interface) - An API is an interface for other programs to interact with your program without giving direct access to their database. A metaphor for this is a bank teller. If you want to get money out of your account, you need to talk to the bank teller (the API) and they will go and get your funds. By having someone between

you and your funds, it keeps others money safe, because you can't steal anyone else's funds.

- Arrays - An array is a container that holds a number of values of a single type. An array can hold for example, test grades [83,94, 86, 91] it can also hold names of your friends [James, Hilary, Steve, Andrew, Tara]. Arrays are good if you need to search through a list, for example, you can retrieve the last friend from the list to take to dinner, or look up what your third test score was.
- Authentication - The process of recognizing a user's identity. It does this by associating an incoming request with a set of identifying credentials (for example, checking whether the password the user put in is the same password on file).
- Class - A class is a definition of the methods and variables in an object, or a sort of template. An object is a specific instance of a class. For example, you can create a class of "circle." Within this class there are certain attributes, like the radius and diameter. Specific objects that share this class would be a Circle X with a radius of 6 inches, or Circle Y with a radius of 4 inches.
- Console - The text entry for code and place where system messages are displayed. Whenever you write out a function in code, the output is displayed on the console.
- Constructor method - A special function that creates initial values for a field. For example, I can say the basic ice cream cone is vanilla ice cream with sprinkles and a waffle cone. So my function to begin with would look like `var ice_cream = (vanilla, sprinkles, wafflecone)`. I can assign other values (sugar cone, chocolate chips, strawberry ice cream) but the constructor method assigns initial values.
- Dialog handlers - A dialog handler is an extension of general handlers. This phrase means how your code is specifically handling text categorized as "dialog."
- Endpoint - URL used to connect to a service on a web server. This shows a website online where it should take data from your computer. Another way to think about it is like a Postal address. The website needs to know the location where it should pick up and drop off mail (a packet of data).
- Ephemeral - A message displayed privately to a user instead of the whole channel

- Event Handlers - This is a general term for an action that happens as a result of an event, or an input. An example would be if a person filled out a form on a website and clicked submit (this would be the event) and then the information from the form is put into a database and then the user is brought to another page (how the event is handled).
- Function - A procedure or routine. When a block of code that contains a function is “called” the computer stops simply outputting script and begins to compute the first line of the function. These instructions might be “add 5 to the hour I input to create a reminder” or “if the number I input is less than forty, add the input number to it.” When the function has finished, the program goes back to where it started with whatever output the function gave in its place. A way to think of a function is a subroutine. Let’s say you want to clean your room. A subroutine would be making your bed. Another subroutine would be organizing your closet.
- Glitch - An online community where people can find apps and websites and then “remix” them by downloading the code, adding a personal spin, and uploading again.
- Method - this is the interface that other classes use to access and modify the data properties of an object. It sets the behavior of a class of objects.
 - Calling Method - Where the initialized method is being used.
 - Called Method - initialization of a method.
- Ngrok - Ngrok is like a tunnel between your personal server to a public endpoint. In this tunnel there are gatekeepers who can catch any nefarious code trying to get on your computer.
- SDK (Software Development Kit) - A good way to think about a kit is a complete package of all the tools to develop a certain application, like the way a model airplane or Legos come with both the pieces as well as the instructions of how to put them together correctly. An SDK (also known as a devkit), will come with libraries, sample code, and relevant documentation to help you develop.
- Server - A server processes requests and delivers data that it is storing to another computer.
- Signing Secret - A variable in Slack that says a user is authorized to make API calls.

