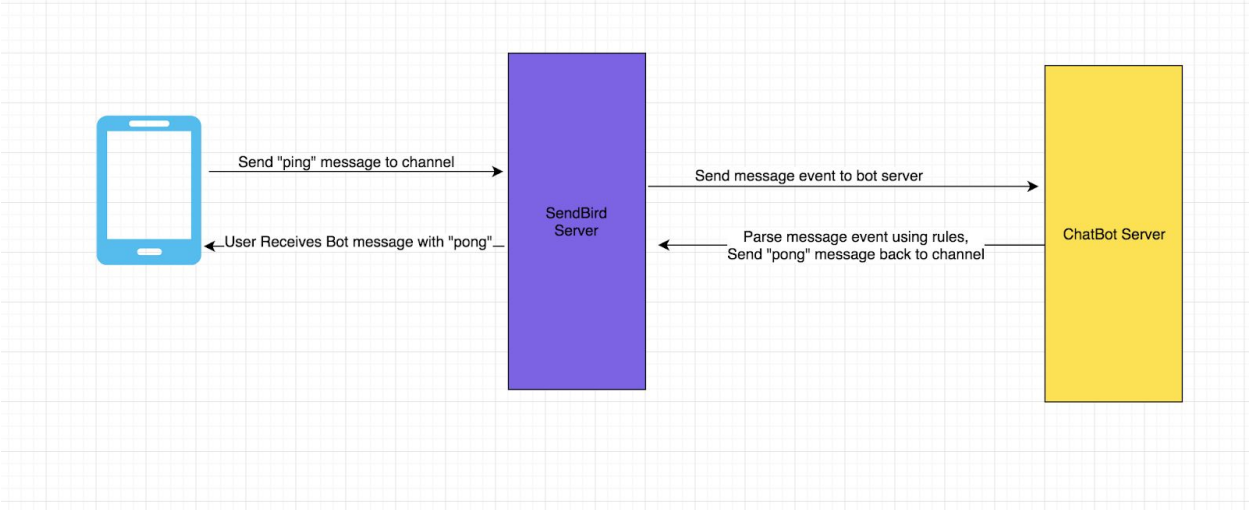
Bot Interface Tutorial: Pong Bot

Date: 2019-05-30
Prepared By: Solutions Engineering

Background

This tutorial will take you through the steps of implementing a simple chat bot which uses the SendBird Platform API to create a bot. This bot will respond with “pong” any time a user in it’s channel sends “ping”.

Message Flow Diagram



Steps covered in this tutorial

By the end of this tutorial you will have a chatbot on AWS Elastic Beanstalk along with a corresponding bot on SendBird ready to respond to user requests.

The steps covered in this tutorial are as follows:

- Create Basic Pong Bot Code
- Deploy Bot on Elastic Beanstalk
- Create SendBird User
- Create SendBird Channel
- Create SendBird Bot
- Join Channel with Bot
- Send Messages to Channel to Trigger Bot Response

Create Basic Pong Bot

For the purposes of this tutorial we have provided sample server code written in Python 3.6 and uses Flask which can be deployed to AWS Elastic Beanstalk. You can use this code as a starting point for your own chat bot.

An explanation of the code is given in the README.md of this repository.
https://github.com/smilefam/simple_pong_chatbot

Please note that you will have to change the **host**, **bot_id** and **api_token** values in the config.py since they will depend on your own Application and Bot values.

Deploy Bot on Elastic Beanstalk

The goal of this tutorial is to simply get you up and running with a chatbot with a publicly accessible endpoint.

The hosting solution that we choose to use in this tutorial is Elastic Beanstalk which allows us to get up and running as fast as possible without worrying about underlying infrastructure. Elastic Beanstalk allows us to let AWS handle the infrastructure and simply get a web server running our sample code.

Note: If you have a preferred cloud/server provider feel free to ignore this Elastic Beanstalk example which is provided as a suggestion for those without prior infrastructure knowledge.

AWS Flask/Python Reference
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create-deploy-python-flask.html

AWS Reference Documentation
https://aws.amazon.com/getting-started/tutorials/deploy-app-command-line-elastic-beanstalk/

Setup the Elastic Beanstalk CLI

For the commands in this tutorial, it is necessary to setup the Elastic Beanstalk CLI. You can follow the AWS documentation on this as there are no steps specific to this tutorial:
https://aws.amazon.com/getting-started/tutorials/set-up-command-line-elastic-beanstalk/

Setup an Elastic Beanstalk Application Directory

Type `eb init` and go through the setup wizard. For the purposes of this tutorial it is fine to simply accept the default options that are given.

```
$ eb init

Select a default region
1) us-east-1 : US East (N. Virginia)
...
(default is 3): 3

Enter Application Name
(default is "sendbird-bot-server-sample"): sendbird-ping-bot
Application sendbird-ping-bot has been created.

It appears you are using Python. Is this correct?
(Y/n): Y

Select a platform version.
1) Python 3.6
...
(default is 1): 1
Do you wish to continue with CodeCommit? (y/N) (default is n): n
Do you want to set up SSH for your instances?
(Y/n): n
```

If you have never setup any AWS service using their CLI before, you may be prompted to provide AWS account authentication which can be retrieved through the IAM section of the AWS Console.

```
You have not yet set up your credentials or your credentials are incorrect
You must provide your credentials.
(aws-access-id): AKIAJOUAASEXAMPLE
(aws-secret-key): 5ZRIrtTM4ciIAvd4EXAMPLEDtm+PiPSzpoK
```

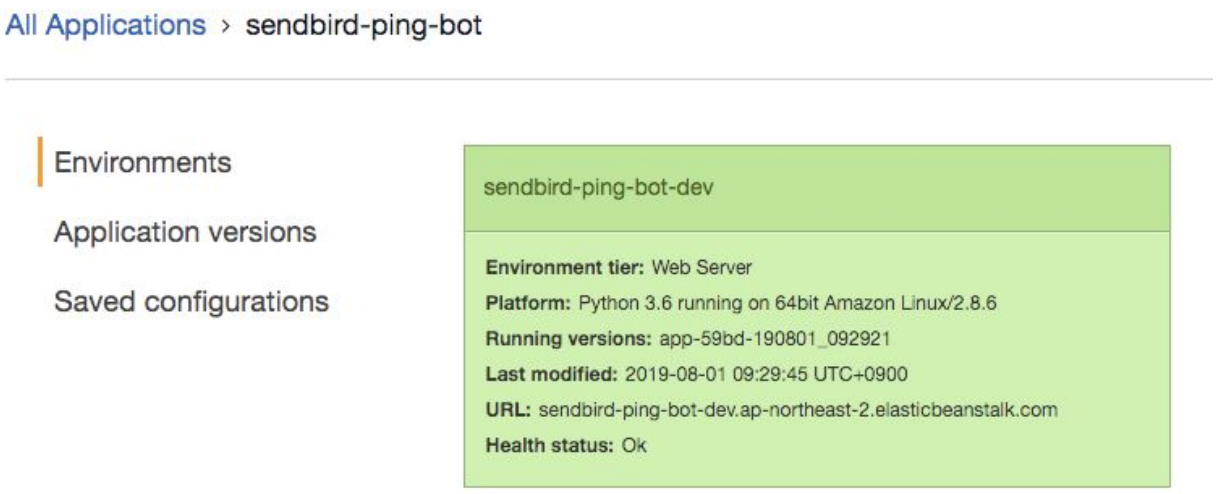
Launch Bot on Elastic Beanstalk

Enter **eb create** in order to launch an Elastic Beanstalk instance, for the purposes of this tutorial you can accept the default options that are suggested by the AWS CLI.

```
$ eb create
Enter Environment Name
(default is sendbird-ping-bot-dev):
Enter DNS CNAME prefix
(default is sendbird-ping-bot-dev):
```

```
Select a load balancer type
1) classic
2) application
3) network
(default is 2):
Creating application version archive "app-59bd-190729_151815".
Uploading sendbird-ping-bot/app-59bd-190729_151815.zip to S3. This may take
a while.
Upload Complete.
...
2019-07-29 06:21:35    INFO    Application available at
sendbird-ping-bot-dev.ap-northeast-2.elasticbeanstalk.com.
2019-07-29 06:21:35    INFO    Successfully launched environment:
sendbird-ping-bot-dev
```

Once you have deployed your bot, you can take a look at [Elastic Beanstalk on the AWS Console](#).



Please take note of the URL of the bot as that will be used as part of the bot_callback_url in the next section.

Setup SendBird Bot

After setting up the needed infrastructure on Elastic Beanstalk, you can setup a corresponding Bot on SendBird using the Bot Interface. The steps below will take you through the creation and configuration of the SendBird Bot which will send commands to your chatbot running on Elastic Beanstalk.

Create SendBird Bot

In order to create a SendBird Bot you can use the Bot Interface - Create Bot Endpoint. Please note that we add the **/bot** endpoint to **bot_callback_url** since our server code for the bot listens for messages on this endpoint.

POST https://api-{application_id}.sendbird.com/v3/bots

cURL

```
curl -X POST \
  https://api-C4D66B91-D13D-48C7-9609-D8B3859A12C8.sendbird.com/v3/bots \
  -H 'Api-Token: <Removed>' \
  -H 'Content-Type: application/json' \
  -F
bot_callback_url=http://sendbird-ping-bot-dev.ap-northeast-2.elasticbeansta
lk.com/bot \
  -F bot_userid=pong_bot \
  -F
bot_profile_url=https://sendbird.com/main/img/profiles/profile_06_512px.png
\
  -F bot_nickname=pong \
  -F is_privacy_mode=false
```

Response

```
{
    "enable_mark_as_read": true,
    "channel_invitation_preference": 0,
    "bot_callback_url":
"http://sendbird-ping-bot-dev.ap-northeast-2.elasticbeanstalk.com/bot",
    "bot": {
        "bot_profile_url":
"https://sendbird.com/main/img/profiles/profile_06_512px.png",
        "bot_token": "0fbd857c321b3e9ffe579c9cd481fc4ce9581575",
```

```
        "bot_nickname": "pong",
        "bot_metadata": {},
        "bot_userid": "pong_bot"
    },
    "show_member": false,
    "is_privacy_mode": false
}
```

Create SendBird User

Create a SendBird User using the Create User Endpoint. You can use any values you wish for the **user_id** but make a note of it for later steps.

cURL

```
curl -X POST \
  https://api-C4D66B91-D13D-48C7-9609-D8B3859A12C8.sendbird.com/v3/users \
  -H 'Api-Token: <Removed>' \
  -H 'Content-Type: application/json' \
  -F user_id=test_user_1 \
  -F nickname=test_user_1 \
  -F
profile_url=https://sendbird.com/main/img/profiles/profile_06_512px.png
```

Response

```
{
    "has_ever_logged_in": false,
    "session_tokens": [],
    "user_id": "test_user_1",
    "access_token": "",
    "discovery_keys": [],
    "is_online": false,
    "last_seen_at": 0,
    "nickname": "test_user_1",
    "profile_url":
"https://sendbird.com/main/img/profiles/profile_06_512px.png",
    "metadata": {}
}
```

Create SendBird Group Channel

Create a SendBird Channel using the Create Channel Endpoint. In this example we add our user **test_user_1** to the channel. We do not add our bot user to the channel as they will be added in the next step.

Save the automatically generated **channel_url** in the response for the following steps.

cURL

```
curl -X POST \
https://api-C4D66B91-D13D-48C7-9609-D8B3859A12C8.sendbird.com/v3/group_chan
nels \
  -H 'Api-Token: <Removed>' \
  -H 'Content-Type: application/json' \
  -F user_ids=test_user_1
```

Response

```
{
    "custom_type": "",
    "is_ephemeral": false,
    "freeze": false,
    "is_created": true,
    "member_count": 1,
    "last_message": null,
    "unread_mention_count": 0,
    "is_discoverable": false,
    "joined_member_count": 1,
    "channel_url":
"sendbird_group_channel_132041699_c81917a5ec2f78ac0145506520fb872ea411dc60"
,
    "channel": {
        "name": "Group Channel",
        "member_count": 1,
        "custom_type": "",
        "channel_url":
"sendbird_group_channel_132041699_c81917a5ec2f78ac0145506520fb872ea411dc60"
,
        "created_at": 1564622077,
        "cover_url": "https://sendbird.com/main/img/cover/cover_06.jpg",
        "max_length_message": -1,
```

```
        "data": ""
    },
    "unread_message_count": 0,
    "is_distinct": false,
    "cover_url": "https://sendbird.com/main/img/cover/cover_06.jpg",
    "members": [
        {
            "state": "joined",
            "user_id": "test_user_1",
            "is_online": false,
            "is_active": true,
            "last_seen_at": 0,
            "nickname": "test_user_1",
            "profile_url":
"https://sendbird.com/main/img/profiles/profile_06_512px.png",
            "metadata": {}
        }
    ],
    "is_public": false,
    "data": "",
    "is_super": false,
    "name": "Group Channel",
    "created_at": 1564622077,
    "is_access_code_required": false,
    "max_length_message": -1
}
```


Join Channel with Bot

Next the SendBird Bot must be added to the channel in order to receive and respond to messages. This can be done using the Bot Interface - Join Channel Endpoint.

POST https://api-{application_id}.sendbird.com/v3/bots/{bot_user_id}/channels

cURL

```
curl -X POST \

https://api-C4D66B91-D13D-48C7-9609-D8B3859A12C8.sendbird.com/v3/bots/pong_
bot/channels \
  -H 'Api-Token: <Removed>' \
```

```
  -H 'Content-Type: application/json' \
  -d '{
    "channel_urls": [
"sendbird_group_channel_132041699_c81917a5ec2f78ac0145506520fb872ea411dc60"
    ]
}'
```

Response

```
{
    "channels": [
        {
            "name": "Group Channel",
            "custom_type": "",
            "channel_url":
"sendbird_group_channel_132041699_c81917a5ec2f78ac0145506520fb872ea411dc60"
,
            "created_at": 1564622077,
            "cover_url": "",
            "data": ""
        }
    ]
}
```

Channel not showing up on Channel List in Sample

The sample app recommended in the next step below does not show channels without messages in the sample list. You can use the following Platform API call to send a test message to the channel so that it is visible in the channel list

cURL

```
curl -X POST \

https://api-C4D66B91-D13D-48C7-9609-D8B3859A12C8.sendbird.com/v3/group_chan
nels/sendbird_group_channel_132041699_c81917a5ec2f78ac0145506520fb872ea411d
c60/messages \
  -H 'Api-Token: <Removed>' \
  -H 'Content-Type: application/json' \
  -d '{"message_type": "MESG", "user_id": "test_user_1", "message":
"test"}'
```

Response


```
{
    "custom_type": "",
    "mentioned_users": [],
    "translations": {},
    "updated_at": 0,
    "is_removed": false,
    "user": {
        "nickname": "test",
        "user_id": "test_user_1",
        "profile_url":
"https://sendbird.com/main/img/profiles/profile_06_512px.png",
        "metadata": {}
    },
    "file": {},
    "message": "test",
    "data": "",
    "type": "MESG",
    "created_at": 1564623311382,
    "mention_type": "users",
    "channel_url":
"sendbird_group_channel_132041699_c81917a5ec2f78ac0145506520fb872ea411dc60"
,
    "message_id": 2580486692
}
```


Send Messages to Channel with User to Trigger the Bot

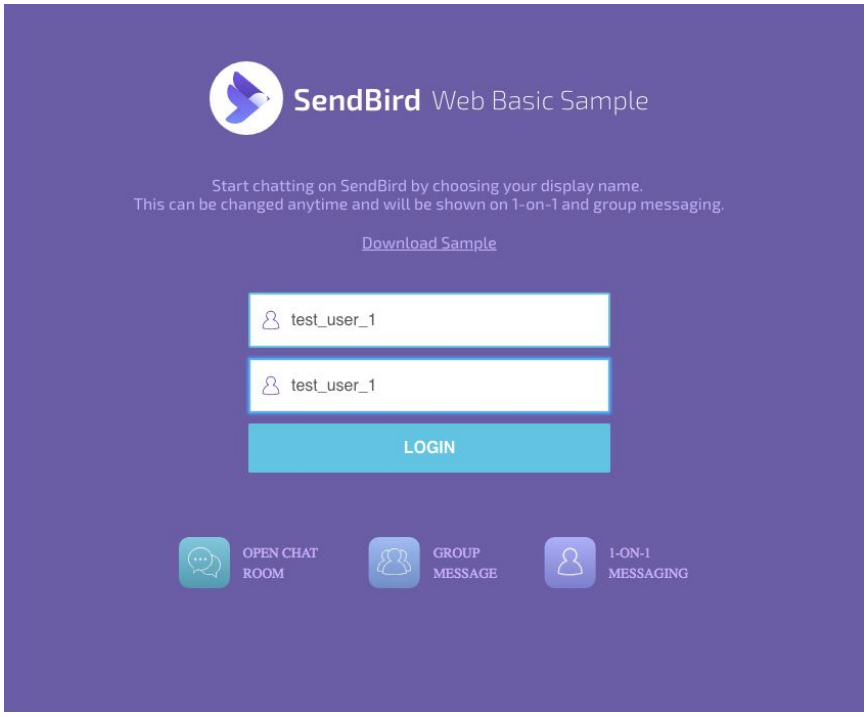
At this point you are ready to send a message to a channel and watch the bot respond.

If you don't already have your own SendBird app up and running, the easiest option is to use one of our sample apps and follow the instructions to replace the default APP_ID with your application's APP_ID.

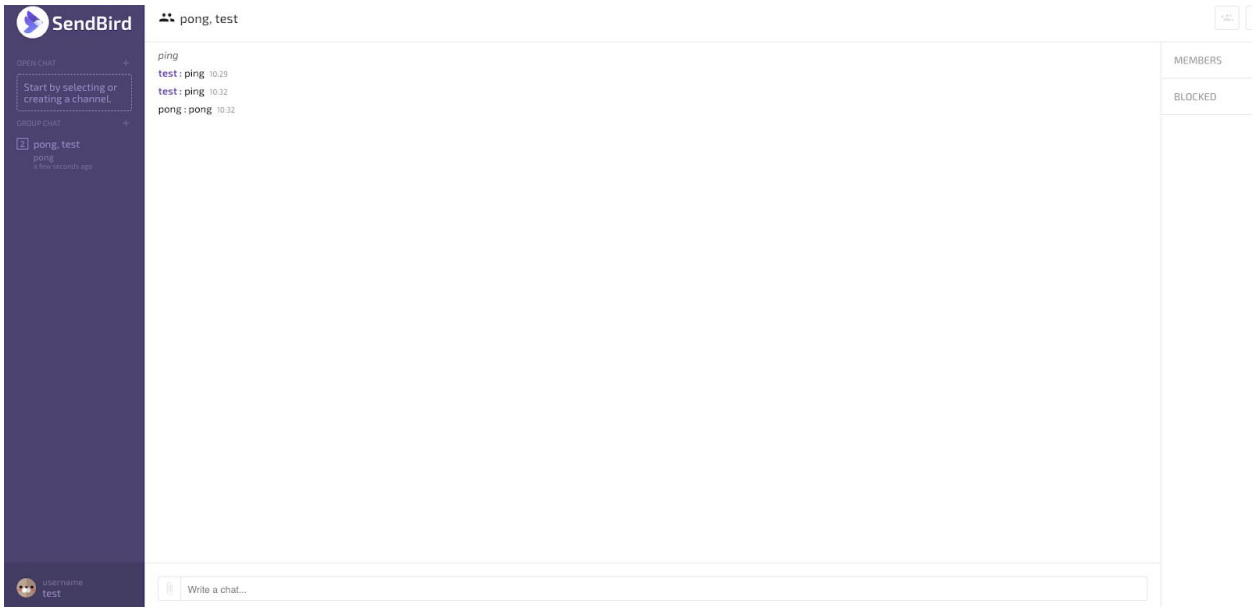
JavaScript Web Basic Sample:
https://github.com/sendbird/SendBird-JavaScript/tree/master/web-basic-sample
Android Sample: https://github.com/sendbird/SendBird-Android/tree/master/basic
iOS Sample: https://github.com/sendbird/SendBird-iOS

Then afterwards you should log in as the SendBird user you created (not the Bot User) so that you can interact with the Bot.

Here in the **JavaScript Web Basic Sample** to login as the user_id created in the previous steps, test_user_1.



Once logged in, send a message “ping” to a channel and the bot responds with a “pong” after receiving the message event from SendBird’s server.



Redeploy changes to Elastic Beanstalk

After you deploy your bot you may want to make changes to your chatbot, such as changing the response or changing the messages it will respond to.

IMPORTANT: "**eb CLI**" deploys the latest committed changes, if you do not make a commit it will not use the latest changes. If you do not want to commit changes you can add changes using "**git add ***" (adds all new changes) and then deploy them using "**eb deploy --staged**".

```
$ git add *
$ eb deploy --staged
Creating application version archive
"app-a355-190801_121448-stage-190801_121448".
Uploading sendbird-ping-bot/app-a355-190801_121448-stage-190801_121448.zip
to S3. This may take a while.
Upload Complete.
2019-08-01 03:14:54    INFO    Environment update is starting.
2019-08-01 03:14:58    INFO    Deploying new version to instance(s).
2019-08-01 03:15:16    INFO    New application version was deployed to
running EC2 instances.
2019-08-01 03:15:16    INFO    Environment update completed successfully.

Alert: An update to the EB CLI is available. Run "pip install --upgrade
awsebcli" to get the latest version.
```

Next Steps

While this is a simple chatbot, you may choose to add more rules to the chatbot. There are cloud chatbot providers as well which you can leverage should your use case require it. In the future we plan to add guides for some of these providers as well which can cover unique requirements for each integration.

For more complex/traffic heavy production use cases we would recommend to use something other than Elastic Beanstalk for better scalability and cost optimization of your chat bot infrastructure. A few possibilities may be moving the chat bot server to AWS EC2/ECS instead.

Google DialogFlow:
https://dialogflow.com/

IBM Watson Chatbot:
https://www.ibm.com/watson/how-to-build-a-chatbot