
Jackalope Documentation

Release 0.2.0

Bryson Tyrrell

May 23, 2017

1	Create the Slack App for Your Team	3
2	Deploying the Slack App	5
2.1	Run from application.py	5
2.2	Testing with ngrok	6
2.3	Environment Variables	6
2.4	Using create_app()	7
3	Docker Example	9
3.1	Connect to a Docker Host	9
3.2	Docker Environment Variables	9
3.3	Build and Run the Containers	10
4	Routes	11
4.1	GET: /	11
4.2	GET: /install	11
4.3	POST: /jamf/<uuid>	11
5	Slack Notifications	13
6	Exceptions	15
	Python Module Index	17

#jackalope

☆ | 2 | 0 | Add a topic

📞 ⚙️ 📄 🔍 Search

Today at 9:41 AM

Computer Added

A new computer has been added!
ID: 1 | Serial Number: C2MS5J3DF0Q1
Computer Name: | User:



Today at 9:45 AM

Mobile Device Un-Enrolled

A mobile device been un-enrolled!
ID: 1 | Serial Number: CLH3QKPZ1FXD
Device Name: | User:



Today at 9:45 AM

Mobile Device Check-In

A mobile device check-in has occurred.
ID: 1 | Serial Number: FCPGDENBH2M3
Device Name: | User:



Today at 9:45 AM

REST API Operation

A REST API operation has been performed.
API Object Type TypeName | **Name:** Name | **ID:** 1
User: bryson.tyrrell | **Action:** GET | **Success?** True



Today at 9:46 AM

Patch Definition Update

Jamf Pro has received a new patch definition update.
[Click here to view the report](#)
Software Title: Microsoft Word | **New Version:** 15.99.0

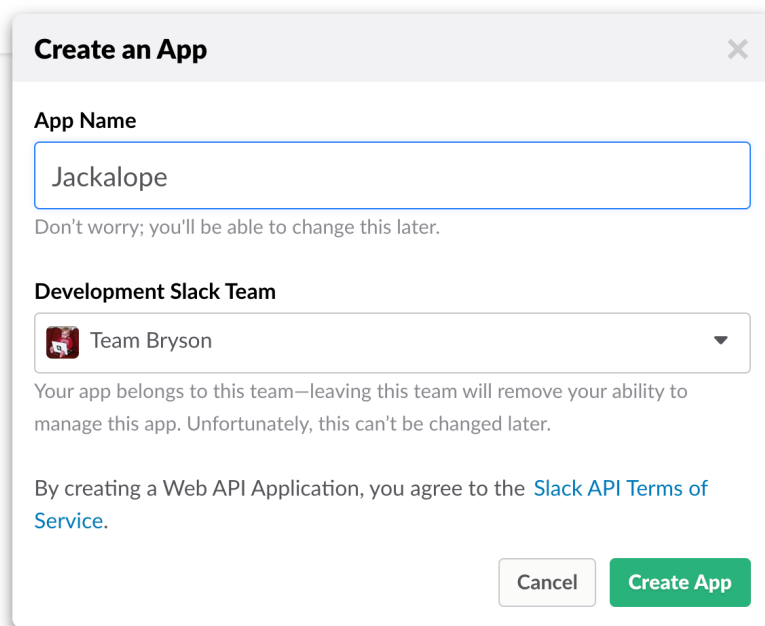


Today at 9:46 AM

Welcome to Jackalope's documentation (still a work-in-progress)!

Create the Slack App for Your Team

Before deploying the app on a server, you will need to create it for your Slack team at <https://api.slack.com/apps>.



Create an App

App Name

Jackalope

Don't worry; you'll be able to change this later.

Development Slack Team

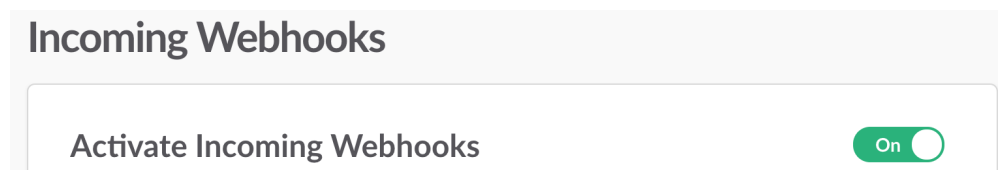
Team Bryson

Your app belongs to this team—leaving this team will remove your ability to manage this app. Unfortunately, this can't be changed later.

By creating a Web API Application, you agree to the [Slack API Terms of Service](#).

Cancel Create App

You will need to enable Incoming Webhooks...



Incoming Webhooks

Activate Incoming Webhooks



...and set a Redirect URL that points to:

https://jackalope.mydomain.org/install

Redirect URLs

You will need to configure redirect URLs in order to automatically generate the Add to Slack button or to distribute your app. If you pass a URL in an OAuth request, it must (partially) match one of the URLs you enter here. [Learn more](#)

Redirect URLs

https://jackalope.mydomain.org/install  

With those two steps complete you will be able to copy the following values for use with the deployed application environment:

- Basic Information/App Credentials/Client ID
- Basic Information/App Credentials/Client Secret

App Credentials

These credentials allow your app to access the Slack API. They are secret. Please don't share your app credentials with anyone, include them in public code repositories, or store them in insecure ways.

Client ID

Client Secret

.....

You'll need to send this secret along with your client ID when making your [oauth.access](#) request.

- Manage Distribution/Share Your App with Your Team/Shareable URL

Shareable URL

Note: You will need to set these into environment variables details in the deployment documentation. See *Environment Variables*.

Deploying the Slack App

Jackalope is written in Python using the Flask framework. This gives a wide variety of options in deploying the application in your environment from installing onto a standalone server, a cloud instance, or within a container.

Note: If you are testing, skip to *Run from application.py* below.

You will need the following components to deploy the application:

- A web server or load balancer to serve traffic over TLS
- A WSGI server (*uWSGI* or *Gunicorn* for example) to run the application code
- A MySQL server

The application will only connect to a MySQL server if all of the required *Environment Variables* have been provided. If not, a local SQLite database will be created within the application directory.

In a cloud instance deployments you can use services such as:

- Amazon Elastic Beanstalk
- Heroku
- Google Apps Engine

Run from application.py

This script creates the root application object from the `jackalope.create_app()` application factory function. If the script has been called from the command line an instance will be launched in a local development server.

```
$ python application.py
```

The development server will be accessible at:

```
http://localhost:5000
```

If the application is being deployed with a WSGI framework, configure the WSGI server to point to the `application.py` file and the `application` object.

Alternatively, the WSGI framework can instantiate and customize an application object using `jackalope.create_app()`.

Testing with ngrok

For testing, you can use the `ngrok` secure tunneling service to expose the application to the internet and access it using both HTTP and HTTPS.

Note: `ngrok` will create randomized subdomains each time you execute the binary (e.g. `4951502d.ngrok.io`). Custom subdomain names are a part of a paid subscription.

Once you have the `ngrok` binary you can create your tunnel.

Expose a local port on your client (5000 in this example):

```
$ ngrok http 5000 --bind-tls true
```

Expose a port on another host from your client (such as a running Docker host):

```
$ ngrok http 192.168.99.100:5000 --bind-tls true
```

When `ngrok` is running you will see the available public endpoints in the window and a stream of traffic logging. Going to `http://127.0.0.1:4040` in your browser will show the web UI and additional details on the requests that are being made through the tunnel.

Environment Variables

Configuration settings are applied at runtime from the environment variables detailed below.

DEBUG

Run the application in debug mode (additional logging).

This setting will set to `True` if any value is provided. To leave this setting disabled do not set it in the environment.

SECRET_KEY

The secret key is a value used to secure sessions with the application.

If a value is not present the application will generate a 16-byte key using `os.urandom()`.

SERVER_NAME

The domain name of the application server.

Example:

```
jackalope.mydomain.org
```

SLACK_CLIENT_ID

The client ID for the Slack application.

This is obtained during the app creation process at:

```
https://api.slack.com/apps
```

SLACK_CLIENT_SECRET

The client secret key for the Slack application.

This is obtained during the app creation process at:

```
https://api.slack.com/apps
```

SLACK_SHAREABLE_URL

The URL provided by Slack for installing the application to channels. This is used for the installation page's "Add to Slack" button that is provided as a part of this application.

This is obtained during the app create process at:

```
https://api.slack.com/apps
```

Note: The following database values are required when connecting Jackalope to a MySQL server. If they are omitted, a SQLite database will be created within the application directory. This is *not* recommended for production deployments.

DATABASE_URL

The URL to the MySQL server with the port.

Example:

```
localhost:3306  
database.mydomain.org:3306
```

DATABASE_NAME

The name of the MySQL database residing on the server.

DATABASE_USER

The username to access the database with.

DATABASE_PASSWD

The password to the user accessing the database.

Using `create_app()`

```
jackalope.create_app()
```

Create the root application object, configure the database object, and register all blueprints from `jackalope.routes`.

Returns Flask application

Return type flask.Flask

Docker Example

The `docker` directory within the Github repository is an example of deploying Jackalope using Docker containers and the `docker-compose` utility.

Warning: This example is not configured for HTTPS traffic - only HTTP over port 80. A production deployment should be configured with a certificate to encrypt traffic using TLS.

`docker-compose` will start three containers (`nginx`, `web` - the application - and `mysql`) on the host. It will also create a volume attached to the `mysql` container to persist the database between container tear-downs (but be warned: if the volume is deleted the database will be lost!).

In your shell/Terminal, `cd` into the `docker` directory before continuing.

Connect to a Docker Host

```
$ eval $(docker-machine env yourhost)
```

Docker Environment Variables

Set the following environment variables within your shell/Terminal:

- `DEBUG`
- `SECRET_KEY`
- `SERVER_NAME`
- `SLACK_CLIENT_ID`
- `SLACK_CLIENT_SECRET`

- SLACK_SHAREABLE_URL
- MYSQL_ROOT_PASSWORD
- MYSQL_DATABASE
- MYSQL_USER
- MYSQL_PASSWORD

Build and Run the Containers

```
$ docker-compose build
$ docker-compose up -d
```

You will be able to access the application at the IP address of the Docker host.

Note: You can use `ngrok` to create a secure tunnel to the Docker host and expose it on the public internet to test with Slack. See [Testing with ngrok](#) for more details.

GET: /

`jackalope.routes.install.root()`
Renders the “Add to Slack” page.

GET: /install

`jackalope.routes.install.install()`
The installation endpoint as set in the “Redirect URLs” for the Slack application under “OAuth & Permissions”.
A `code` parameter will be provided by Slack to this URL once an installing user has authorized the application for a channel.
Using the `code` along with the application’s set `SLACK_CLIENT_ID` and `SLACK_CLIENT_SECRET` values, the application will make a POST to `https://slack.com/api/oauth.access` to obtain an `access_token` and other details for the channel installation.
Upon an `ok` response to the request, the details will be saved to the database, a UUID will be generated for receiving Jamf Pro webhooks, and a success message displayed in the Slack channel.
If the Slack channel already exists in the database its details will be updated with those from the response.

POST: /jamf/<uuid>

`jackalope.routes.jamfpro.jamf_webhook(jamf_uuid)`
The receiver endpoint where `jamf_uuid` is the auto-generated UUID for an installed Slack channel.
Inbound webhooks must be in JSON format or a 400 error will be returned.

If a supported webhook event has been received it will be formatted into a Slack message via `jackalope.routes.jamfpro.webhooks.webhook_notification()` and sent via `jackalope.slack.send_notification()`.

Parameters `jamf_uuid` (*str*) – The generated UUID for the installed Slack channel.

Raises `SlackChannelLookupError`

Raises `JSONNotProvided`

Returns HTTP 204 success response.

Slack Notifications

These functions handle the processing of Jamf Pro webhook events to Slack notifications.

`jackalope.routes.jamfpro.webhooks.webhook_notification(webhook)`

Takes a Jamf Pro webhook event object and returns a formatted Slack message from the details if it is in the supported webhook events list.

If the webhook event is not supported `None` will be returned.

Parameters `webhook` – Jamf Pro webhook JSON as dictionary.

Returns Formatted Slack message.

Return type dict or None

`jackalope.routes.jamfpro.webhooks._message(text, title, title_link=None, color='gray', fallback_text=None, image=None, fields=None)`

Create a Slack formatted message to use with `jackalope.slack.send_notification()`.

Parameters

- **text** (*str*) – The main text to display in the message.
- **title** (*str*) – The title of the message.
- **title_link** (*str*) – An optional URL to pass that will convert the title into a clickable link.
- **color** (*str*) – The color to display in the sidebar of the message. Must be of a value in the `_colors` dictionary or will default to `gray`.
- **fallback_text** (*str*) – Alternative text to display in place of the provided `text` value. If not submitted this will be set to the value of `text`.
- **image** (*str*) – The filename of an image located in `/static/images/` to link to with the message. If not submitted this will be set to `general_64.png`.
- **fields** (*dict*) – A dictionary of keyword values to populate the optional `fields` attribute of the Slack message.

`jackalope.slack.send_notification(url, message)`

Send a formatted Slack message to a channel's inbound webhook.

Parameters

- **url** (*str*) – The URL for a Slack channel's inbound webhook.
- **message** (*dict*) – A formatted Slack message generated by `jackalope.routes.jamfpro.webhooks._message()`.

```
class jackalope.routes.errors.JackalopeException  
    Base Jackalope Exception  
  
class jackalope.routes.errors.JSONNotProvided  
    A valid JSON body was not provided with a request.  
  
class jackalope.routes.errors.SlackChannelLookupError  
    Exception raised when performing a lookup of an installed Slack channel.
```


a

application, 5

j

jackalope.config, 6

jackalope.routes.errors, 15

Symbols

`_message()` (in module `jackalope.routes.jamfpro.webhooks`), 13

A

`application` (module), 5

C

`create_app()` (in module `jackalope`), 7

E

environment variable

- `DATABASE_NAME`, 7
- `DATABASE_PASSWD`, 7
- `DATABASE_URL`, 7
- `DATABASE_USER`, 7
- `DEBUG`, 6
- `SECRET_KEY`, 6
- `SERVER_NAME`, 6
- `SLACK_CLIENT_ID`, 6
- `SLACK_CLIENT_SECRET`, 7
- `SLACK_SHAREABLE_URL`, 7

I

`install()` (in module `jackalope.routes.install`), 11

J

`jackalope.config` (module), 6

`jackalope.routes.errors` (module), 15

`JackalopeException` (class in `jackalope.routes.errors`), 15

`jamf_webhook()` (in module `jackalope.routes.jamfpro`), 11

`JSONNotProvided` (class in `jackalope.routes.errors`), 15

R

`root()` (in module `jackalope.routes.install`), 11

S

`send_notification()` (in module `jackalope.slack`), 13

`SlackChannelLookupError` (class in `jackalope.routes.errors`), 15

W

`webhook_notification()` (in module `jackalope.routes.jamfpro.webhooks`), 13