Channel Islands
CALIFORNIA STATE UNIVERSITY

Voyager 2 S.A.V.E.I.T.

Samuel Decanio & Vlad Synnes • Professor Michael Soltys • COMP 499 Capstone

Introduction

This project is intended to provide a method of tracking information, such as an IP address and other packet information, about an individual who clicks a link to an image that is hosted on our project's server. This automated process keeps a log of web traffic information that is parsed to actively update a database to reflect these requests. The end-user is intended to interact with this project via our web application to avoid unintentional mishaps that may impair the function of the software. However, an admin may access project contents via command-line interface to make further changes that may not be permitted through our front-facing web interface. To accommodate the dependencies of this software, an installation script is provided to create an quick and easy set-up of a fresh EC2 instance.

End-User Flow

This end user interface is designed to present a graphical view of the current photo and event records. With it, you will also have the ability to edit event notes and upload new images.

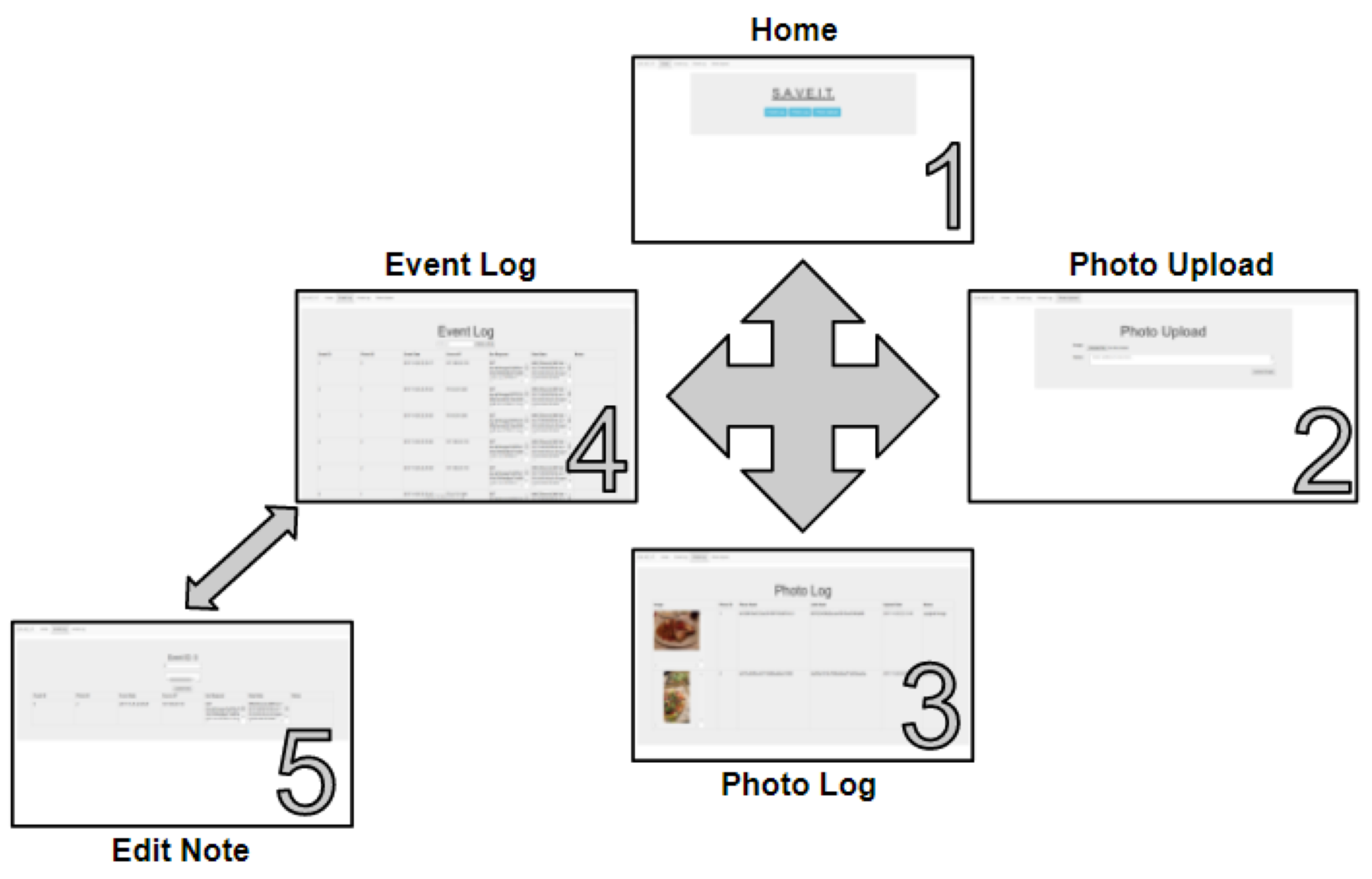
1. The **home pages** gives the user the ability to navigate to the photo log, event log or photo upload page using either the navigation bar or buttons.

2. The **photo upload** page allows you to browse your file system for an image, you may also enter a note that will be associated with the photo log record of your uploaded image.

3. The **photo log** page shows the user all of the current photos that have been uploaded to the database. The column which displays the link_hash contains a hyperlink to the page on which the corresponding image is available. This can easily by copied by right clicking and selecting "Copy Link Address".

4. The **event log** page shows the user the all of the current events that have been logged into the database. The user may also look up a specific event by typing in a given record id into the text box provided and clicking the "Retrieve Note" button.

5. The **note edit** page returns the table results of the users record id input from the event log page, if the event id was not found the table will be empty. To edit the note, the user enters his/her notes into the text area and clicks "update note". If the user wishes to clear the notes on the given record, he/she can simply leave the text area blank and click "update note". If the user wishes to not make any edits, he/she can navigation to another page using the navigation bar.



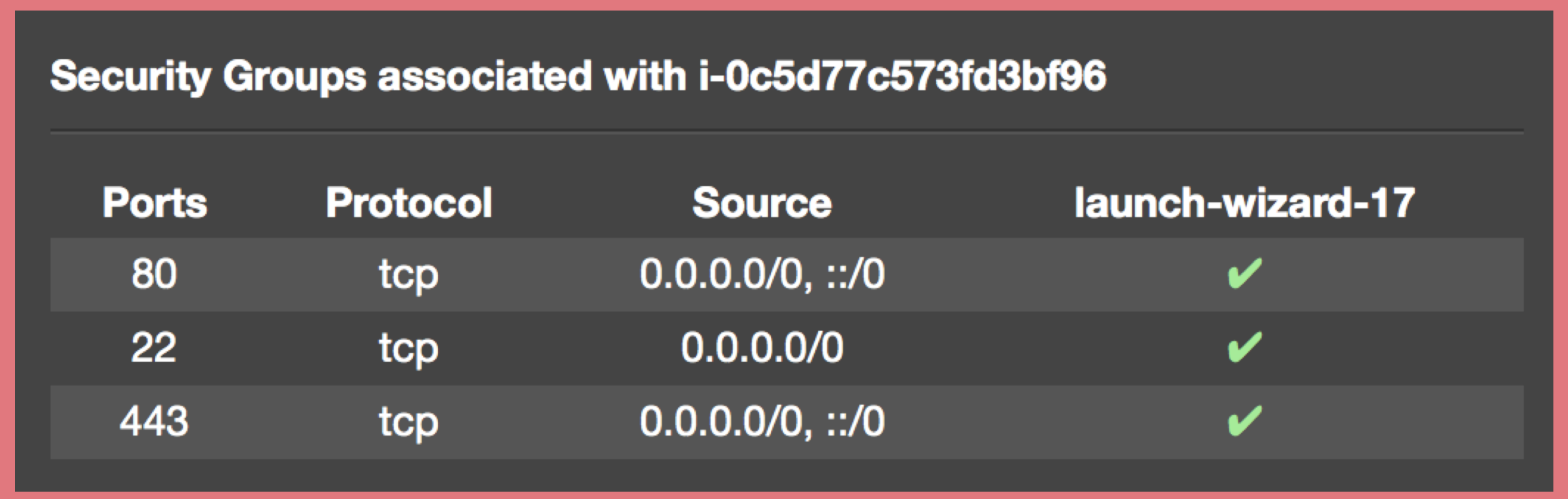
Provisioning

In order to allow our project to be easily deployable and portable we created an installation shell script allowing for quick creation and configuration of EC2 instances. There are numerous steps the script accomplishes in order to correctly configure the instance.

1. Installs the LAMP Server Stack (httpd24, php70, mysql-server, php70-mysqlnd)
2. Downloads our project files from a private GitHub repository
3. Creates the Voyager2 database and associated tables
4. Sets up the correct file structure
5. Installs Python library: MySQLdb
6. Installs Python library: Scapy
7. Installs tcpdump package
8. Creates cron job responsible for running .PCAP parser script
9. Output the public IP Address from which images will be served

Technologies Used

• **AWS EC2** - Amazon Web Services virtual computer instance. We used this instance to host our application from. Amazon's virtual instances provided a good way for us to host our front end web application as well as our backend scripts and database while being easy to configure and recreate upon any development mishaps that ruined the server. We are using an Amazon Linux instance, t2.micro, details: amzn-ami-hvm-2017.09.1.20171103-x86_64-gp2 (ami-32d8124a), with the following inbound rules:



Security Groups associated with i-0c5d77c573fd3bf96

Ports	Protocol	Source	launch-wizard-17
80	tcp	0.0.0.0/0, ::/0	✓
22	tcp	0.0.0.0/0	✓
443	tcp	0.0.0.0/0, ::/0	✓

• **Tcpdump** - Tcpdump is a command line packet analyzer. In our project we used it in order to capture incoming TCP requests to our server which were then stored in .PCAP files to later be analyzed. We leveraged the options of this tool in order to create a new .PCAP file on a set time interval which is named with the date on which it is created.

• **Python** - We chose to use python because of its aptitude towards rapid development and prototyping as well as its diverse range of available libraries. This enabled us to create many iterations quickly as well as test ideas and different implementations in a timely manner.
  • **Scapy** - A python module we used to extract, process and manipulate tcpdump files (.PCAP). We used Scapy specifically to extract the GET request, raw packet contents, date and URL.

• **Cron** - Cron is a time based job scheduler. We created a cron job to run our Python script which was responsible for parsing and archiving our (.PCAP) files on a scheduled basis.

• **MySQL** - MySQL was used to store our event logs as well as the records corresponding to uploaded photos. It being a relational database allowed us to create corresponding connections between photo records and the events generated through their viewing.

• **PHP** - PHP was used intermittently throughout our GUI implementation to communicate with and make changes within our database.

• **HTML & CSS** - HTML and CSS was used for both our structure and styling of our web application.

• **Bash** - Bash was utilized in order to create the installation script which is used to download, install, and configure all required files and dependencies in order for our project to run on a fresh EC2 instance.

Design Flow

1. To begin, an image must be uploaded via the image upload web page photo_upload.php.

2. The image is then processed with the image_upload.php script that generates the record in the photo_log table of our database.

3. It also renames the image to its link_hash, and moves it into the /images folder.

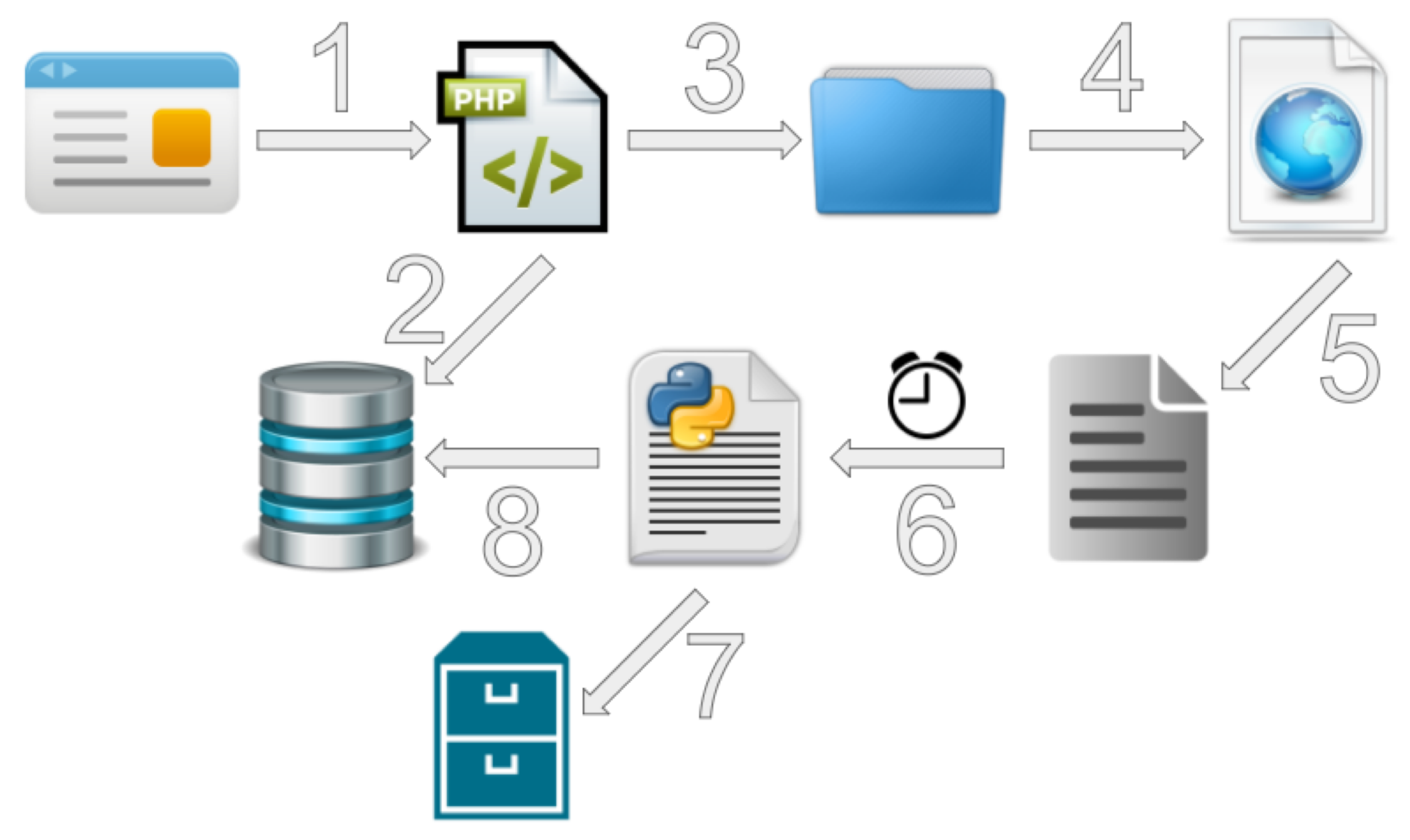
4. The image is now available to be viewed on the following URL: http://<IP_ADDRESS>/saveit/images/<link_hash>.jpg

5. The tcpdump utility running in the background will be logging all incoming TCP traffic, such as requests for the hosted images. When your image is viewed that traffic will be captured in a .PCAP file which is stored (with a name of the date time it was generated) in the /tcpdump directory. The naming convention of the .PCAP files using standard date formatting is as follows: tcpdump_%F_%H:%M:%S.pcap.

6. A scheduled cron job running the parser.py script will read all .PCAP files contained in the directory and parse them for GET Requests and other information.

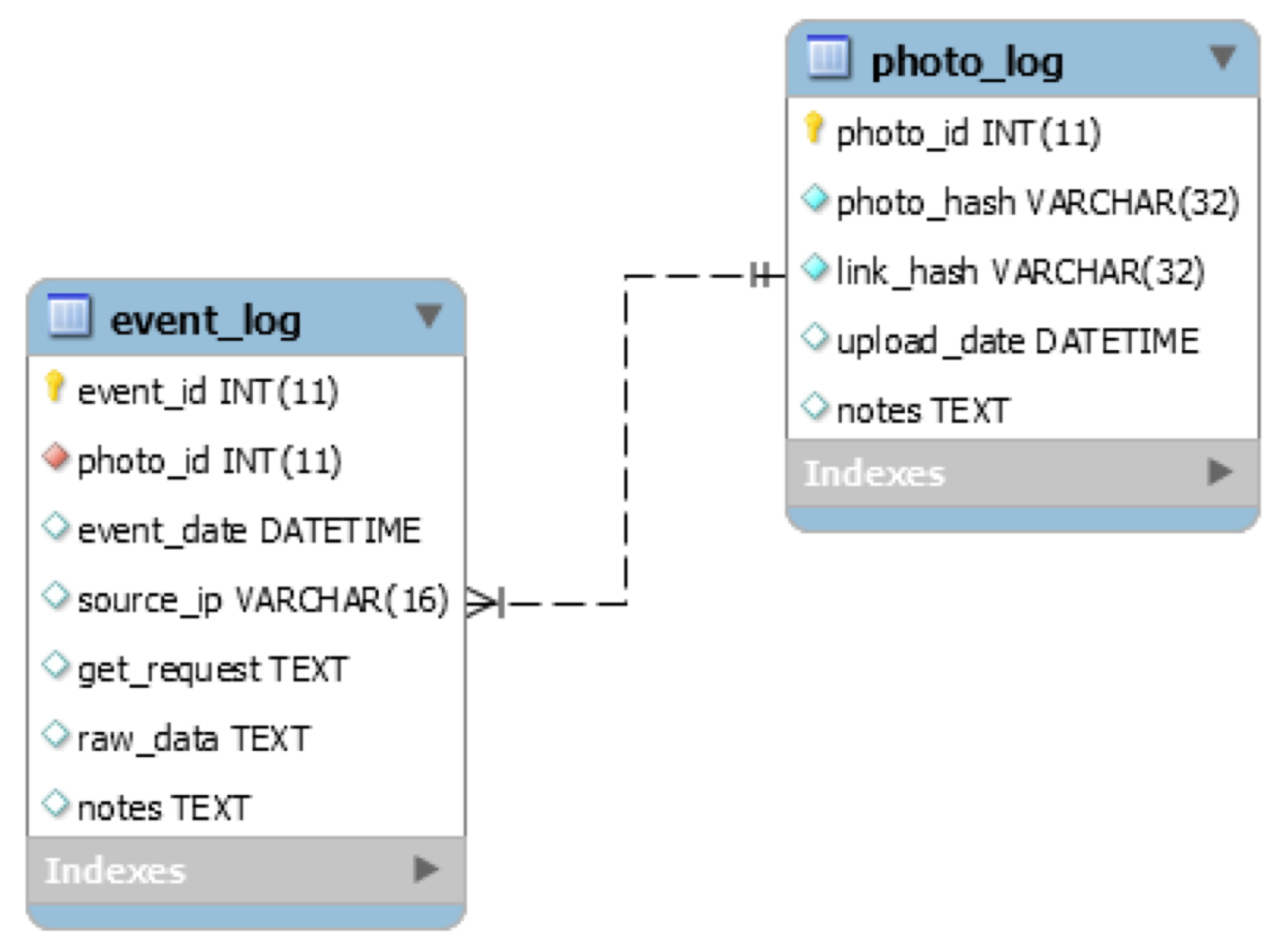
7. Once a .PCAP file has been completely parsed the script then moves it to the /archive directory for long term storage. The files are not deleted incase they need to be examined at a later date.

8. When requests within the .PCAP files are found they are then logged in the event_log table of the database which stores information about the request as well as creates a relation to which specific image was view.



Database Structure

• **Voyager2** - The database in which we store records associated with uploaded images and event associated with those images.
  • **photo_log** - The photo log table contains all the information related to each image that has been uploaded to our server.
  • **event_log** - The event log contains all recorded information from our parsed .PCAP file along with a photo_id to accurately associate it to the corresponding image.


photo_log
photo_id INT(11)
photo_hash VARCHAR(32)
link_hash VARCHAR(32)
upload_date DATETIME
notes TEXT
Indexes
event_log
event_id INT(11)
photo_id INT(11)
event_date DATETIME
source_ip VARCHAR(16)
get_request TEXT
raw_data TEXT
notes TEXT
Indexes