Design Document

Project: Software Discovery Tool

Mentees: Indranil Mandal, Divya Goswami
Mentor: Elizabeth K. Joseph

Introduction

Software Discovery Tool(SDT) is a tool for discovering open source software for zArchitecture/s390x for any Z operating system from any source, any repo, anywhere, in one place.

Problem Definition

Currently SDT only supports RHEL/SLES/Ubuntu based systems. Our mentorship task is to add z/OS support, thus extending the usability of SDT for IBMZ too. Our task also includes basic UI improvements and finally separate SDT as a separate project (which was previously working as a fork of Package Distro Search(PDS)).

Requirements

For production:

1. For SDT to be live usable, a server is needed
2. Storage Space of minimum: 50MB disk space
3. Server should have python, apache2 installed.

For development:

1. All the specifications are mentioned in the Installation guideline.

Milestones

1. Basic UI improvements
2. Text explaining what the website is
3. Linking to the Open Mainframe Project
4. Adding a Contribute link to GitHub repo
5. Support for z/OS
6. Backend JSON file for open source z/OS applications
7. Improvements to FAQ, including how to Bring Your Own JSON file back ends
8. Improvements to Documentation
9. Stretch goal: API for querying JSON back ends
10. Stretch goal:Backend JSON files for Debian, openSUSE,ClefOS, etc
11. Stretch goal: CI/CD
12. Stretch goal: Hosted instance of tool

All this work will be done on:
https://github.com/openmainframeproject/software-discovery-tool

Architecture

SDT has been forked from PDS, an existing tool to search software based on linux distribution. It uses the Flask web framework written in Python to serve the json data from the data folder to the web page. This Flask application is deployed in an Apache web server using wsgi.

Detailed Design

Currently, SDT has 4 components:

1. A distro data repository: (Repo Link)
2. A Flask application layer server
3. Apache2 web server.
4. Basic Jquery cdn UI

The main source code lies in the directory: src
It contains the following directories:

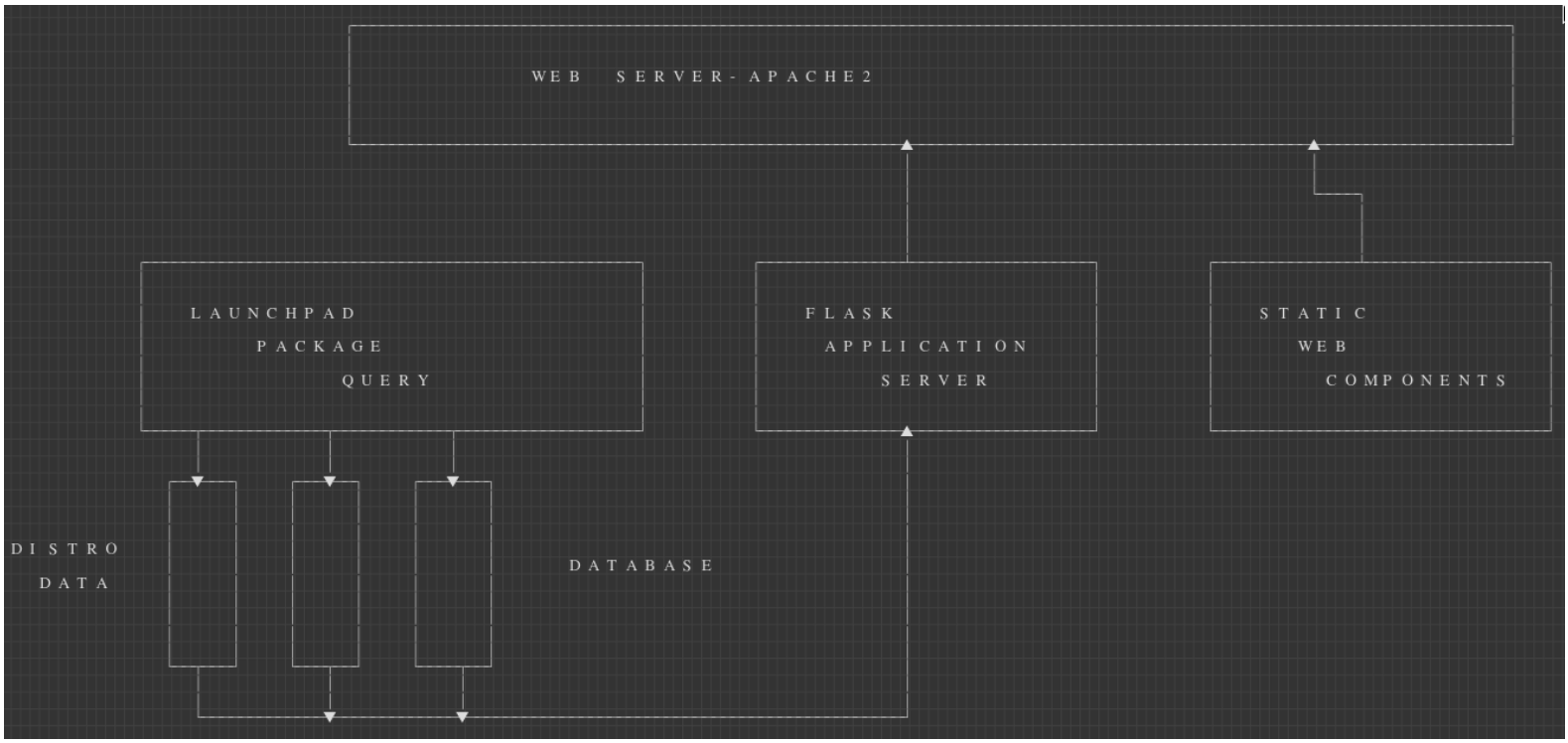
1. **classes**
   This contains the package_search class which contains functions to load and prepare supported distros, package data as well as function to search packages.
2. **bin**
   package_build.py - as the name suggests, is capable of pulling package json files from PDS and also pull fresh lists for Debian and ClefOS incase OMP data directory doesn't contain the latest lists.
   config_build.py - is the automation script which scans the distro_data directory for new files and updates the SUPPORTED_DISTROS object to map new json files. It also updates any previous PDS files that was there in the directory and deletes the cached_data.json file so that the server can be reloaded without any errors.
3. **config**
   config.py- This contains paths and other variables which are imported into tests modules and required by pytest
   supported_distros.py - This file contains the SUPPORTED_DISTROS object that will be read by the controller to reflect json files in the UI.
   sdt.conf- config file for flask application
4. **scripts**
   Uses launchpad to get package data for supported distributions from archive and creates json files for them
5. **static**
   This directory contains the major UI components (i.e. js, css, html) for view,home and faq regions
6. **templates**
   This directory is intended to preserve the basic structure of any page henceforth built for SDT to save time on developing separate site structure.
7. **tests**
   This directory contains 2 major test scripts that checks
   - whether the json files are available or not

- Whether the 2 currently available webpages (home and faq) are safely accessible or not

And remaining 2 files:

1. SDT.wsgi
   Without this file, SDT will fail to operate at the application server layer. This is the init file for establishing a flask application server.
2. main.py
   This is the flask server file that has info about all the servable directories. Currently contains home, faq, Supported Distros and Search Package directories.

The following diagram is the visualization of the whole tool architecture:



Our core project will not be making changes to this architecture, but rather building upon it through the addition of support for z/OS and additional JSON files.

One of our stretch goals is creating an API for querying, which will alter the architecture, but we will submit a revised architecture plan if we are able to start work on this goal.