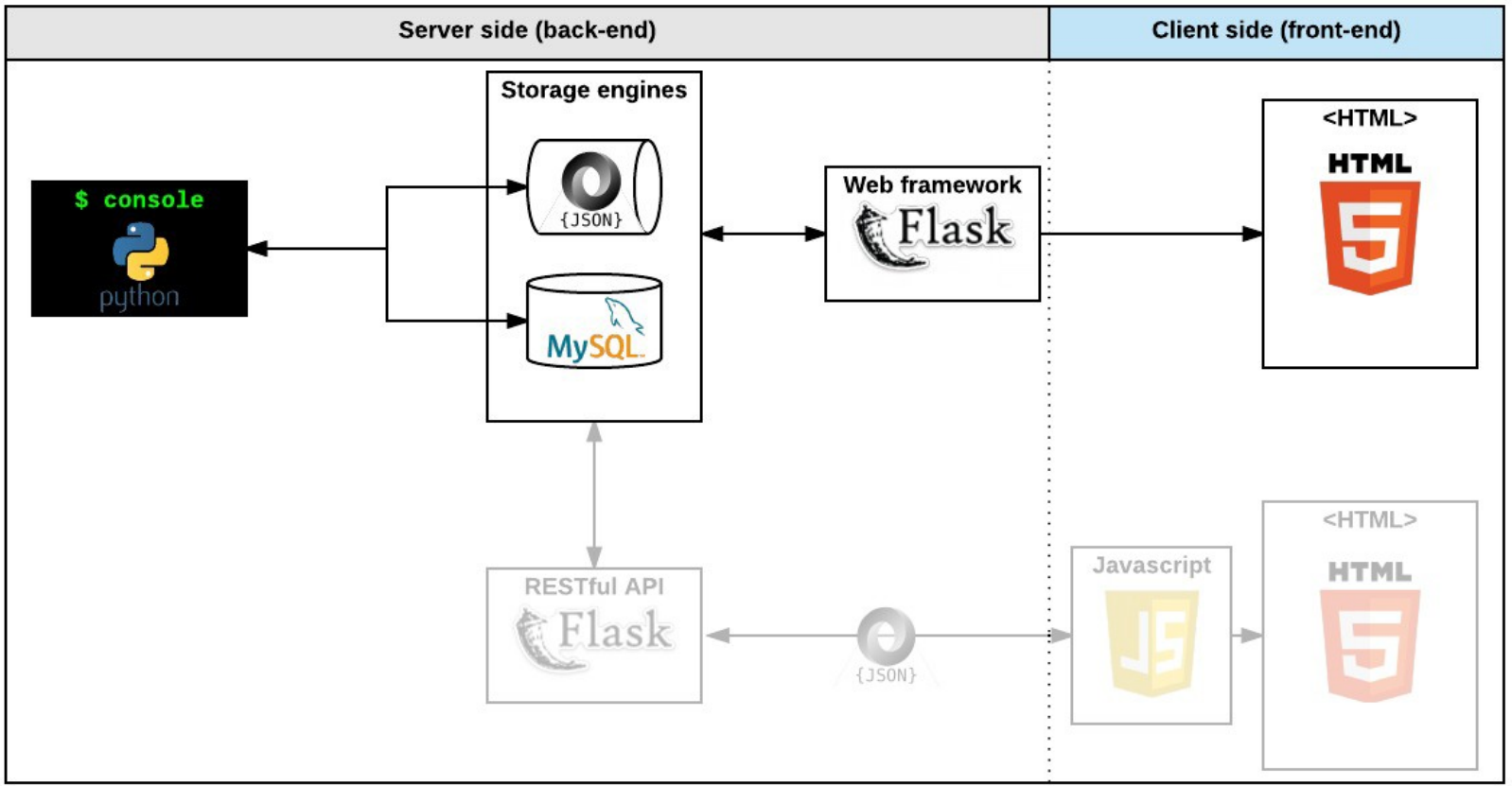
FLASK

- Flask is a small and lightweight **Python web framework** that provides useful tools and features that make creating web applications in Python easier.
- Flask is built on the **Werkzeug WSGI toolkit** and **Jinja2 template engine**.
- Werkzeug werkzeug German noun: "tool". Etymology: werk ("work"), zeug ("stuff")
- **WSGI – Web Server Gateway Interface.**
- Werkzeug is a comprehensive **WSGI web application library.** which implements requests, response objects, and other utility functions. This enables building a web framework on top of it.
- Jinja2 is a full-featured template engine for Python. A web templating system combines a template with a certain data source to render dynamic web pages.

Web Server Gateway Interface(WSGI)

- It is a specification that defines a standard interface between web servers and Python web applications or frameworks.
- WSGI enables web servers and web applications to communicate with each other in a consistent and interoperable way.
- WSGI is a Python standard described in detail in PEP 3333.
- WSGI applications (meaning WSGI compliant) can be stacked.
- Those in the middle of the stack are called **middleware** and must implement both sides of the WSGI interface, application and server.
- A web server that is WSGI compliant only receives the request from the client, pass it to the application and then send the response returned by the application to the client.
- It does nothing else.
- All the other details must be supplied by the application or middleware.
- https://wsgi.readthedocs.io/en/latest/#


Server side (back-end)
Client side (front-end)
Storage engines
{JSON}
MySQL
$ console
python
Web framework
Flask
<HTML>
HTML
RESTful API
Flask
{JSON}
Javascript
<HTML>
HTML

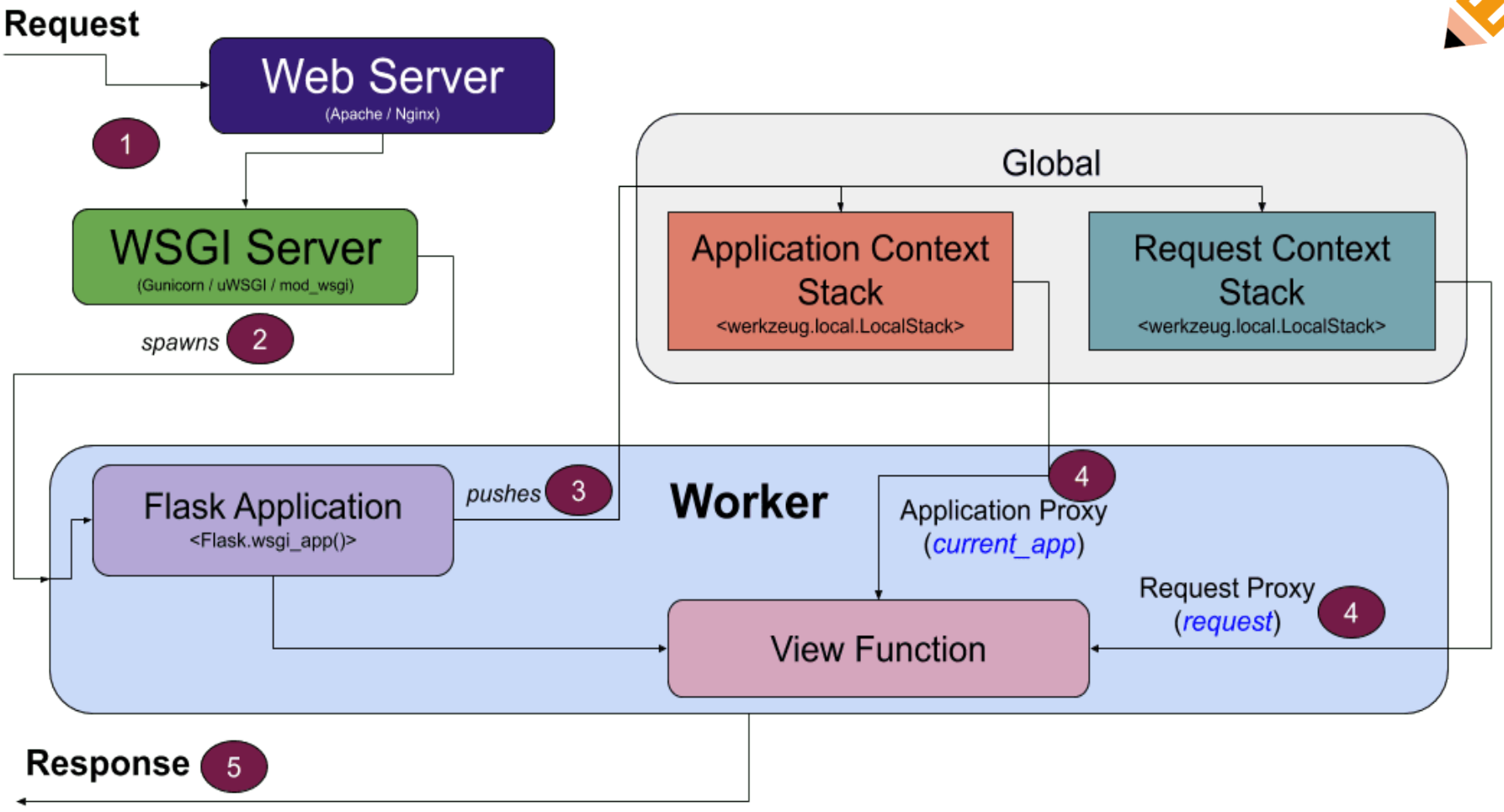
Werkzeug WSGI includes

1) An interactive debugger that allows inspecting stack traces and source code in the browser with an interactive interpreter for any frame in the stack.
2) A full-featured request object with objects to interact with headers, query args, form data, files, and cookies.
3) A response object that can wrap other WSGI applications and handle streaming data.
4) A routing system for matching URLs to endpoints and generating URLs for endpoints, with an extensible system for capturing variables from URLs.
5) HTTP utilities to handle entity tags, cache control, dates, user agents, cookies, files, and more.
6) A threaded WSGI server for use while developing applications locally.
7) A test client for simulating HTTP requests during testing without requiring running a server.

Jinja2

- Jinja2 is a full-featured template engine for Python.
- It has full unicode support, and an optional integrated sandboxed execution environment.
- It works by combining a template with a certain data source to render dynamic web pages.
- It combines a template (the layout of the page) with data (the specific information you want to show) to create a dynamic web page.
- Jinja2 templates are text files that contain a mix of static content and placeholders, called template variables or expressions, enclosed in double curly braces { { ... } } or using the { % ... % } syntax for control statements like loops and conditionals.

How does flask fit into the request response cycle of our webapp?


Request
Web Server
(Apache / Nginx)
1
WSGI Server
(Gunicorn / uWSGI / mod_wsgi)
spawns
2
Global
Application Context Stack
<werkzeug.local.LocalStack>
Request Context Stack
<werkzeug.local.LocalStack>
Flask Application
<Flask.wsgi_app()>
pushes
3
Worker
4
Application Proxy
(current_app)
Request Proxy
(request)
4
View Function
Response
5

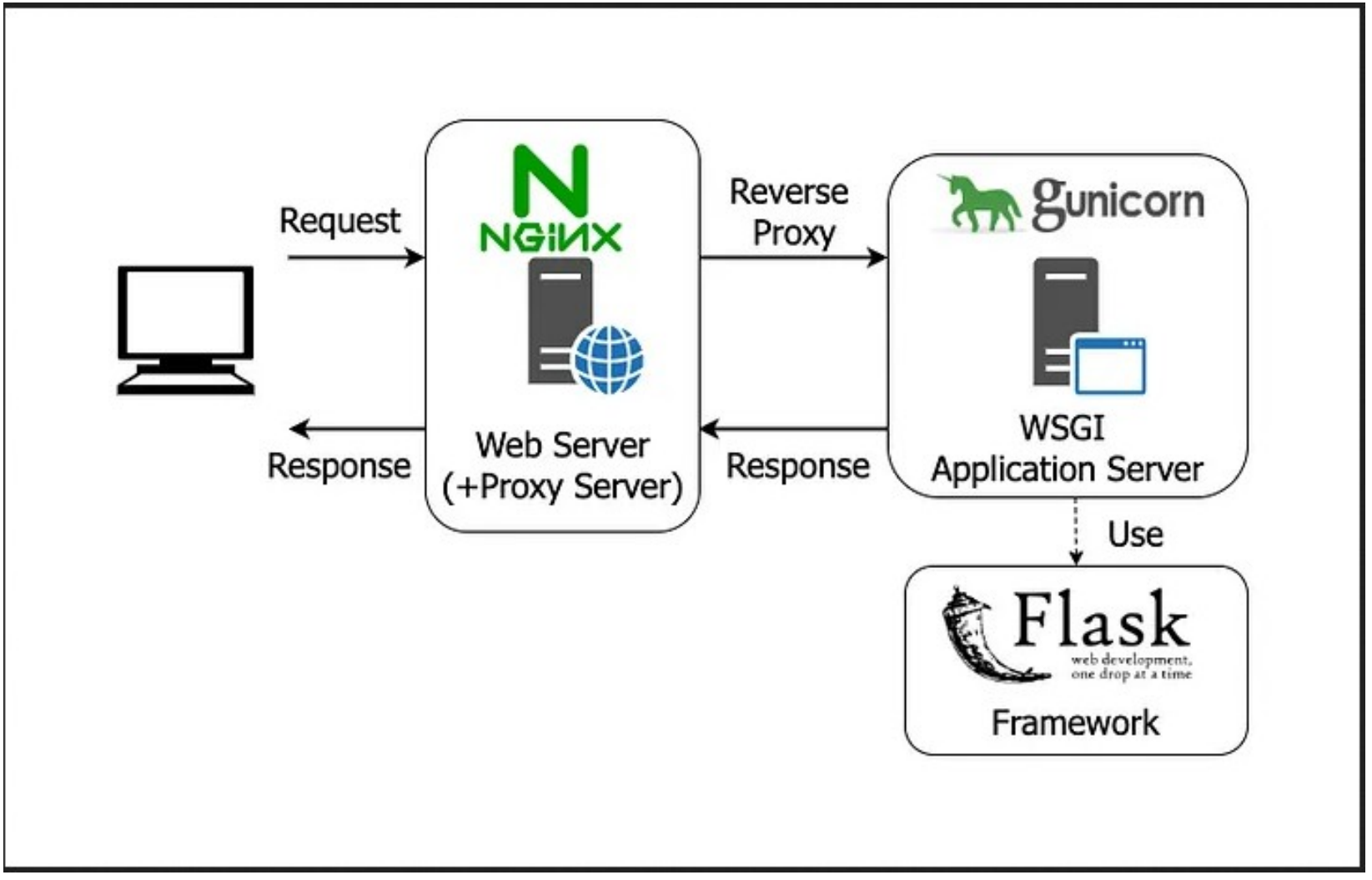
Why use Flask ?

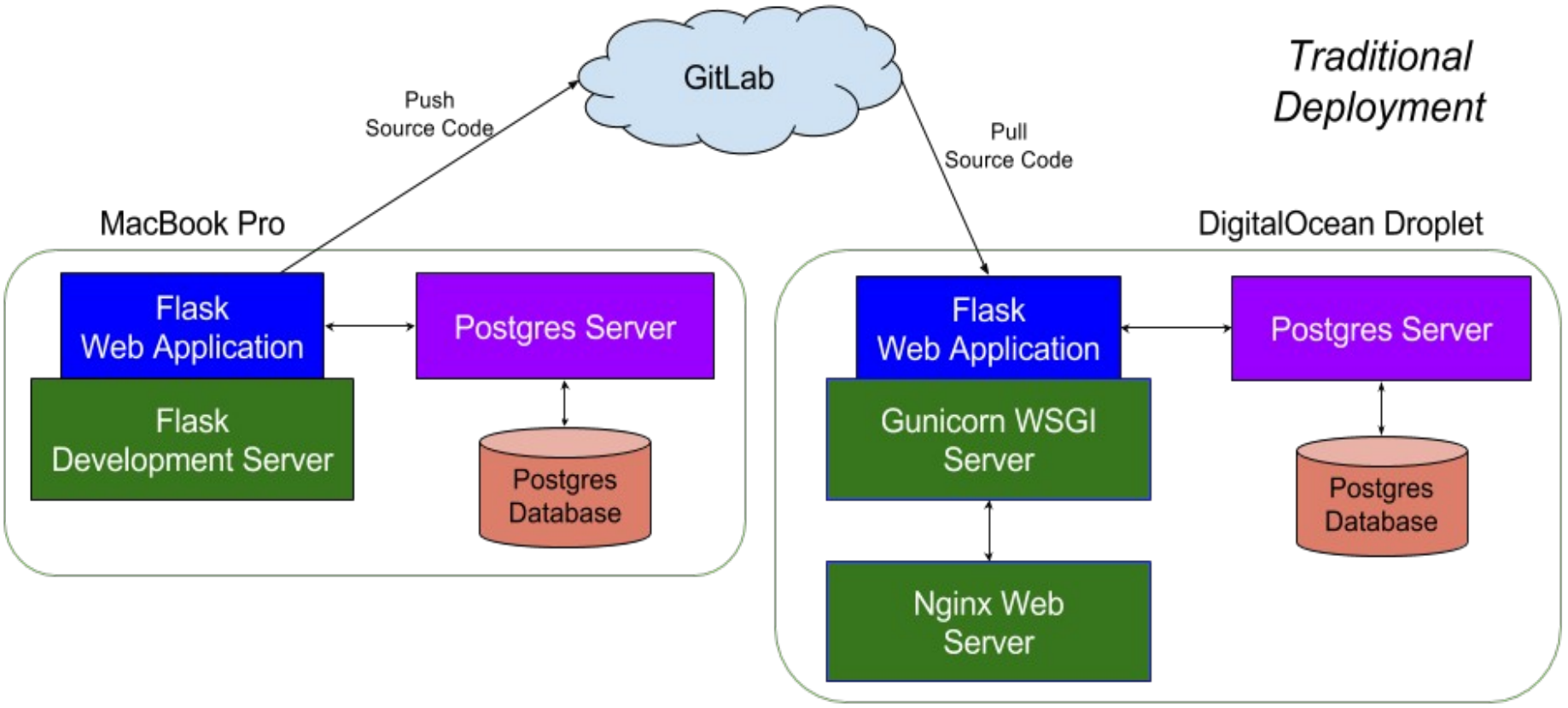
- Flask uses Python and is lightweight. It consumes minimum resources to get the work done.
- Flask has a shallow learning curve It's very easy to learn once you have a sound knowledge of Python.
- Flask enables rapid prototyping of your app and works very efficiently for small applications.
- Among the important features of Flask are a built-in web server and debugger, unit testing support, RESTful request dispatching, secure cookies, WSGI compliance, Unicode support and good documentation.
- Example of sites using Flask nclude:
- Red Hat, Airbnb, Netflix, Lyft, Reddit, Uber, and Samsung. Other big names include Pinterest, Twilio and LinkedIn.

Flask in a production environment

- Flask can be used to implement an Model-View-Controller (MVC) pattern of web framework:
- Flask (controller), SQLite (model), Bootstrap (view). Jinja2 is popular for rendering HTML templates and is part of the view.
- Some commonly used extensions with Flask:
  - **Flask-Cors:** A Flask extension for handling Cross-Origin Resource Sharing (CORS), making cross-origin AJAX possible.
  - **Flask-User:** Customizable user authentication, user management, password recovery, and more.
  - **Flask-PyMongo:** This bridges Flask and PyMongo. Provides some convenient helpers. PyMongo is a MongoDB driver.
  - **Flask-SQLAlchemy:** Database abstraction layer and Object Relational Mapper (ORM) popular for Flask apps.

- Flask comes with a simple built-in server which runs on the default port 5000 commonly used for development but not recommended for production because it doesn't scale well.
- For production, many cloud providers document procedures on how to deploy Flask on their platforms.
- If you wish to deploy Flask on your own, mod_wsgi is an essential module for Apache server.
- For Nginx web servers it will typically serve static files and the rest will be handed over to Gunicorn server, which becomes the WSGI entry point for the Flask application.
- In fact, Nginx can reverse proxy a HTTP request to any WSGI-compliant server including Apache, Gunicorn or uWSGI.


Request
NGINX
Reverse
Proxy
gunicorn
Response
Web Server
(+Proxy Server)
Response
WSGI
Application Server
Use
Flask
web development,
one drop at a time
Framework


Traditional Deployment
GitLab
Push
Source Code
Pull
Source Code
MacBook Pro
Flask
Web Application
Flask
Development Server
Postgres Server
Postgres
Database
DigitalOcean Droplet
Flask
Web Application
Gunicorn WSGI
Server
Nginx Web
Server
Postgres Server
Postgres
Database