ALX Web infrastructure design

TASK 0-simple_web_stack

specifics about this infrastructure:

1. **Server**:
    - A server is a computer or a machine responsible for processing requests and serving content to users over the internet. In this scenario, we have one server that will handle all incoming requests and host our website.
2. **Domain Name**:
    - A domain name (in this case, "foobar.com") is a human-readable address that points to the server's IP address (8.8.8.8). It serves as a user-friendly way to access the website instead of remembering the server's IP address.
3. **DNS Record**:
    - The "www" in www.foobar.com is a DNS (Domain Name System) record. Specifically, it's a CNAME (Canonical Name) record that points to the root domain (foobar.com) or the server's IP address. This record helps route requests made to www.foobar.com to the correct server.
4. **Web Server (Nginx)**:
    - The web server, in this case, Nginx, is responsible for receiving incoming HTTP requests from users and serving static content, like HTML, CSS, and images. It acts as a reverse proxy, passing dynamic requests to the application server.
5. **Application Server**:
    - The application server hosts the application codebase. It processes dynamic requests, executes the code, and generates the appropriate responses. In a LAMP stack, it's typically where PHP, Python, or other server-side code is executed.
6. **Application Files (Codebase)**:
    - The application files consist of your website's source code. This includes HTML templates, server-side scripts, and other assets that the application server uses to generate web pages dynamically.
7. **Database (MySQL)**:
    - The database, in this case, MySQL, stores and manages the website's data. It can include user information, content, and any other data needed for the website to function. The application server communicates with the database to retrieve or update data as required.
8. **Communication with User's Computer**:
    - The server uses the HTTP protocol to communicate with the user's computer. When a user accesses www.foobar.com, their computer sends an HTTP request to the server. The server processes the request, retrieves data from the database (if necessary), and sends an HTTP response back to the user's computer, which displays the website in their web browser.

Issues with this infrastructure:

1. **Single Point of Failure (SPOF)**:
    - The entire infrastructure relies on a single server. If that server experiences hardware or software issues, the website will be unavailable. To mitigate this, you can consider redundancy and failover solutions, such as load balancing and backup servers.
2. **Downtime During Maintenance**:
    - When performing maintenance, such as deploying new code that requires the web server to be restarted, the website may experience downtime. To minimize downtime, you can implement rolling deployments and use a load balancer to direct traffic to other servers while one is being updated.
3. **Scaling Challenges**:
    - This infrastructure may struggle to handle a large influx of traffic. To scale, you can introduce load balancers and add more servers to distribute the load. Additionally, you can consider optimizing the database and application code to handle more concurrent users efficiently.