Algorithm:

1. Initialization

1.1 Import necessary libraries (tornado, os, koji, json).

1.2 Define constants (URL parts, build trigger, build target, Koji server URL).

1.3 Get the absolute path of the certicficates using the OS library and save it in the variables.

2. Create handler Class

2.1 Define a *koji_build* method to trigger a Koji build using a provided URL.

2.1.a Start the client session using koji server URL

2.1.a.1 O/P-----> It will return the client session ID

2.1.b Use session id to login to the server using ssl_login

2.1.b.1 Arguments to ssl_login-----> CLIENTCERT, CLIENTCA, SERVERCA

2.1.c Call the builder using sessionid.build

2.1.c.1 Arguments-------> URL to the source directory of package, Build target

2.2 Define a *post* method to handle incoming requests

TRY BLOCK:

2.2.a Read the JSON data from the body and convert it into python dictionary.

2.2.a.1 O/P------->Object to the dictionary

2.2.b Create a json file and dump the data into the file (Like – sample.json)

2.2.c Open the created json file and get the data object

2.2.c.1 Arguments--------> file object

2.2.c.2 O/P----------> data object

2.2.d Parse the data and get the commit message

2.2.e Check if the commit message contains the BUILD_TRIGGER string.

2.2.e.1 **If found**: Extract the commit ID and URL from the JSON data.

2.2.e.1.1 Construct the final URL for the Koji build.

2.2.e.1.2 Call *koji_build* method to trigger the build.

2.2.e.2 If not found: Log a message indicating the absence of the trigger string.

EXCEPT BLOCK

Handle potential JSON decoding errors by setting a 400 status code and returning a response indicating invalid payload.

3. Create Application

3.1 Define a function *make_app* to create a Tornado web application with a single route (/) that maps to the *handler* class.

4. Run the Server (Main function)

4.1 Create a Tornado application instance.

4.2 Start the server on a specific port.

4.3 Start the Tornado IOLoop to keep the server running, waiting for incoming requests.

Overall Algorithm Flow:

1. The server listens for POST requests on the defined port.
2. When a POST request arrives, the *handler.post* function is called.
3. The JSON data containing commit information is parsed and extracted.
4. The commit message is checked for the presence of the BUILD_TRIGGER string.
5. If found, the Koji build is triggered using the extracted commit URL and target build environment.
6. If not found, the server logs a message and continues listening for new requests.

.