```clojure
(defn file-handler [{:keys [queue conn] :as ctx} {:keys [^File file kind] :as event}]
(f/insert-file conn {:name (.getName file) :processed (Date.)})))
```

```
(defn weapons-query-handler [{{:strs [name]} :params :keys [conn] :as request}]
{::jdbc/connection    {:connection-uri "jdbc:h2:mem:mem_only"}
```

```
(timbre/debug (str "Detected change to file: " (.getName file)))
```

```
(defn add-weapons-handler [{:keys [params conn] :as request}]
(let [[name & weapons] (map cs/trim (cs/split line #",")))]
```

```
::jdbc/init                        {:conn (ig/ref ::jdbc/connection)}
```

```clojure
(let [data (map (fn [[k v]] {:name k :weapon v}) params)]
::hawk/watch                  {:groups [{:paths   ["example"]
```

```
(defmethod ig/init-key ::jdbc/init [_ {:keys [conn]}]
;(timbre/debug "Checking for new items in queue...")
```

```
(defn queue->dsdb [{:keys [queue-name queue dsdb]}]
(when-some [task (dq/take! queue queue-name 10 nil)]
::scheduling/job          {:job          #'queue->dsdb
```

```
{:data {:coercion   reitit.coercion.spec/coercion
```

```
::durable/queues              {:delete-on-halt? true
```

```
(timbre/debug "Putting data into datascript")
```

```
:schedule  {:in [5 :seconds] :every :second}
```

```
:dsdb          (ig/ref ::datascript/connection)}
```

```
::web/server                {:host     "0.0.0.0"
```

```
:conn     (ig/ref ::datascript/connection)
```

```
(d/transact! dsdb [(line->record @task)])
```

```
(ok (with-out-str (pp/pprint request)))
```

```
;We were able to compose the routes here
```

```clojure
(defn files-handler [{:keys [sql-conn]}]
(ok (w/weapons @conn (keys params))))))
```

```
(cs/ends-with? (.getName file) ".csv"))
```

```
(ok (f/all-processed-files sql-conn)))
```

```
(constantly (not-found "Not found"))))
```

```
(timbre/debug "Adding data to queue.")
```

```
["/add_weapon" add-weapons-handler]])
```

```
:queue          (ig/ref :durable/queues)
```

```
(when (and (#{:modify :create} kind)
:sql-conn (ig/ref ::jdbc/connection)
```

```
[["/weapons" weapons-query-handler]
:conn   (ig/ref ::jdbc/connection)}
```

```
:queue (ig/ref ::durable/queues)
```

```
["/files" {:get files-handler}]])
```

```
(dq/put! queue :my-queue line)))
```

::datascript/connection w/schema

```
(w/everybodys-weapons @conn))))
```

```
:middleware [params/wrap-params
```

```
(with-open [r (io/reader file)]
[basic-routes weapons-routes]
```

```
[["/echo" {:get echo-handler}]
```

```
(defonce sys (create config))
```

```
(defn echo-handler [request]
middleware/wrap-format]}}))
```

```
(assoc :weapon weapons))))
```

```
(doseq [line (line-seq r)]
```

```clojure
(defn line->record [line]
:handler #'file-handler}]
```

```
(w/weapons @conn [name])
```

```
:directory          "/tmp"}
```

```
(d/transact! conn data)
```

```
(dq/complete! task))))
```

```
:handler  #'handler}})
```

```
:queue-name :my-queue
```

```
(cond-> {:name name}
(def weapons-routes
;Routes - All data
```

```
(ring/ring-handler
(def basic-routes
;Global handler
```

```
(f/setup conn))
```

```
(.exists file)
```

```
(.isFile file)
```

```
(seq weapons)
```

```
(ring/router
:port     3000
```

```
(def handler
(def router
(def config
```

;Handlers

```
(if name
```

;Router

router

```
ctx)
```

(**ok**

(do

```
(defn file-handler [{:keys [queue conn] :as ctx} {:keys [^File file kind] :as event}]
(f/insert-file conn {:name (.getName file) :processed (Date.)})))
```

```clojure
(defn weapons-query-handler [{{:strs [name]} :params :keys [conn] :as request}]
{::jdbc/connection      {:connection-uri "jdbc:h2:mem:mem_only"}
```

```
(timbre/debug (str "Detected change to file: " (.getName file)))
```

```
(defn add-weapons-handler [{:keys [params conn] :as request}]
(let [[name & weapons] (map cs/trim (cs/split line #","))]
::jdbc/init                    {:conn (ig/ref ::jdbc/connection)}
```

```clojure
(let [data (map (fn [[k v]] {:name k :weapon v}) params)]
::hawk/watch                    {:groups [{:paths   ["example"]
```

```
(defmethod ig/init-key ::jdbc/init [_ {:keys [conn]}]
;(timbre/debug "Checking for new items in queue...")
```

```
(defn queue->dsdb [{:keys [queue-name queue dsdb]}]
(when-some [task (dq/take! queue queue-name 10 nil)]
::scheduling/job                {:job              #'queue->dsdb
```

```
{:data {:coercion   reitit.coercion.spec/coercion
```

```
::durable/queues                    {:delete-on-halt? true
```

```
(timbre/debug "Putting data into datascript")
```

```
:schedule  {:in [5 :seconds] :every :second}
```

```
:dsdb                 (ig/ref ::datascript/connection)}
```

```
::web/server            {:host     "0.0.0.0"
```

```
:conn    (ig/ref ::datascript/connection)
```

```
(d/transact! dsdb [(line->record @task)])
```

```
(ok (with-out-str (pp/pprint request))))
```

```
;We were able to compose the routes here
```

```clojure
(defn files-handler [{:keys [sql-conn]}]
(ok (w/weapons @conn (keys params))))))
```

```
(cs/ends-with? (.getName file) ".csv"))
```

```
(ok (f/all-processed-files sql-conn)))
```

```
(constantly (not-found "Not found"))))
```

```
(timbre/debug "Adding data to queue.")
```

```
["/add_weapon" add-weapons-handler]])
```

```
:queue          (ig/ref ::durable/queues)
```

```
(when (and (#{:modify :create} kind)
:sql-conn (ig/ref :jdbc/connection)
```

```
[["/weapons" weapons-query-handler]
:conn   (ig/ref ::jdbc/connection)}
```

```
:queue  (ig/ref ::durable/queues)
```

```
["/files" {:get files-handler}]])
```

```
(dq/put! queue :my-queue line)))
```

::datascript/connection w/schema

```
(w/everybodys-weapons @conn))))
```

```
:middleware [params/wrap-params
```

```
(with-open [r (io/reader file)]
[basic-routes weapons-routes]
```

```
[["/echo" {:get echo-handler}]
```

```clojure
(defonce sys (create config))
```

```
(defn echo-handler [request]
middleware/wrap-format]}}))
```

```
(assoc :weapon weapons))))
```

```
(doseq [line (line-seq r)]
```

```clojure
(defn line->record [line]
:handler #'file-handler}]
```

```
(w/weapons @conn [name])
```

```
:directory          "/tmp"}
```

```
(d/transact! conn data)
```

```
(dq/complete! task))))
```

```
:handler  #'handler}})
```

```
:queue-name :my-queue
```

```
(cond-> {:name name}
(def weapons-routes
;Routes - All data
```

```
(ring/ring-handler
(def basic-routes
;Global handler
```

```
(f/setup conn))
```

```
(.exists file)
```

```
(.isFile file)
```

```
(seq weapons)
```

```
(ring/router
:port        3000
```

```
(def handler
(def router
(def config
;Handlers
```

```
(if name
```

;Router

router

ctx)

(ok

(do