U0LJU20SJ : <@U0E0XL064> no, from what I understand it is about the macro expansion of the specs:```
src/service/routing/spec.clj:40:17: suspicious-expression: and called with 1 args. (and x) always returns x. Perhaps there are misplaced parentheses?

src/service/routing/spec.clj:44:22: suspicious-expression: and called with 1 args. (and x) always returns x. Perhaps there are misplaced parentheses?

src/service/routing/spec.clj:38:20: constant-test: Test expression is always logical true or always logical false: 2 in form (if or__5058__auto__ or__5058__auto__ (clojure.core/or 0))

src/service/routing/spec.clj:38:20: constant-test: Test expression is always logical true or always logical false: nil in form (if nil (clojure.core/inc nil) 2)

src/service/routing/spec.clj:38:20: constant-test: Test expression is always logical true or always logical false: nil in form (if or__5058__auto__ or__5058__auto__ (clojure.core/or
```

U0E0XL064 : Well, I guess eastwood is not clojure.spec compliant yet. You may add an issue to <https://github.com/jonase/eastwood/issues> ?
U0E0XL064 : similar issues seem to exist already: <https://github.com/jonase/eastwood/issues/207>
U0LJU20SJ : jumm I was hoping to be wrong. The question is rather is eastwood at fault or clojure.spec? because those expanded expressions do look suspicious
U1B0DFD25 : Is `clojure.core/hash` always non-negative?
U060FKQPN : no
U060FKQPN : it uses the full int range
U1B0DFD25 : Thanks
U3J7HSKNC : this might be a dumb question - it likely is - but what is the best way to define a spec in `clojure.spec` for an argument that is a function of arity 1?
U3J7HSKNC : is that a thing?
U3J7HSKNC : Something that could be used to spec a function like `take!` from `core.async`:
```
(defprotocol ReadPort
  (take! [port fn1-handler] "derefable val if taken, nil if take was enqueued"))
```

U3J7HSKNC : for `fn1-handler`
U068SUJNT : How could I combine two images?
U1B0DFD25 : What's the pattern in Integrant/Component REPL development to pick up an edited configuration file on reset? Do you make it a component in the system? I'm building my system object according to the config so I can't do that.
U06FTAZV3 : <@U1B0DFD25> I tend to read the config file each time and merge the result into my system map. Using Aero to read the EDN file I then do something like this: `(merge-with merge system config)`.
U06FTAZV3 : With a component like this:
```
(defrecord Datomic [uri]
  component/Lifecycle
  ;; ...
)
```

And a config file like this:

```
{:datomic {:uri "datomic:<mem://my-app>"}}
```

U06FTAZV3 : And a system map like this:
```
(component/system-map :datomic (map-&gt;Datomic {}))
```
```

U06FTAZV3 : Integrant does things slightly differently I believe?
U06FTAZV3 : Looks like Integrant will reload your config from this:
<https://github.com/weavejester/integrant#configurations>
U1B0DFD25 : <@U06FTAZV3> Integrant config is the equivalent to Component's system, so it won't just re-read my external config file.
U06FTAZV3 : Yeah, from a quick look at the Integrant README, there is no explicit system map in Integrant.
U1B0DFD25 : That's what I like about it, the system map is data.
U06FTAZV3 : Component's system map is data too. :slightly_smiling_face:
U06FTAZV3 : If you've got a `config` somewhere as per the README, and reloading is done via clojure.tools.namespace, I'd expect changes to the file to show up.
```
(def config
  (ig/read-string (slurp "config.edn")))
```

U06FTAZV3 : Can't say I've used Integrant before though. Sorry.