U61KCTX8S : that's far more elegant than my solution
U0567Q30W : Sure, no problem - thanks!
U3L6TFEJF : I'd love to hear more about this as well, I'm going to try out Integrant in a project in the coming weeks and I've been wondering the same thing
U0BKWMG5B : So there are two ways to do this.
U0BKWMG5B : First, if you're testing a single key, then you can use `init-key` directly and pass the stubbed/mocked connections.
U0BKWMG5B : For example, say you have a key `:foo.handler/user` that takes a database option. You could test it with:```
(ig/init-key :foo.handler/user {:db (->StubbedDatabase)})
```

U0BKWMG5B : Because the database isn't accessed directly, but via protocol methods, we can create a stubbed or mocked version with the same interface. Shrubbery is a test tool that streamlines this process.
U0BKWMG5B : If you're testing the configuration in a wider context, then you can take advantage of keyword inheritance.
U0BKWMG5B : For example, say you had a configuration like:```
{:duct.database.sql/hikaricp
 {:jdbc-url ...}
 :foo.handler/user
 {:db #ig/ref :duct.database.sql/hikaricp}}
```

U0BKWMG5B : One feature of Integrant is that you can reference *derived* keys, so you could write the above as:
U0BKWMG5B : ```{:duct.database.sql/hikaricp
 {:jdbc-url ...}
 :foo.handler/user
 {:db #ig/ref :duct.database/sql}}
```

U5ZAJ15P0 : <@U0BKWMG5B> oh, so in that case you wouldn't swap any implementation, you would simply instruct the system to use a different implementation based on the config (different derived key)
U0BKWMG5B : So if you want to stub out the key directly, then change the database key to a fake one that derives from the same base:```
(derive :duct.database.sql/fake :duct.database/sql)
```

U5ZAJ15P0 : Thank you, I'll try both of those approaches :slightly_smiling_face: I had a follow up question but you answered it. It was going to be: "in your talk you mention how you can instantiate two systems with different configurations, but how can I instantiate two systems with different implementations?"
U0BKWMG5B : Right: you could update the configuration to replace the real database with a fake one:```
{:duct.database.sql/fake
 {:jdbc-url ...}
 :foo.handler/user
 {:db #ig/ref :duct.database/sql}}
```

U5ZAJ15P0 : From what I gather the answer to this would be "you only have one implementation per keyword; you just use a different config"
U0BKWMG5B : Right. Just take the base config and alter it with `assoc`. Or use `duct.core/merge-configs` to merge in new options.
U0BKWMG5B : It effectively amounts to the same thing.
U0BKWMG5B : You *could* also use `with-redefs` to redefine the `init-key` multimethod, but since that's not thread-safe I'd advise avoiding that route.
U5ZAJ15P0 : Thanks for the explanation! So the gist is that I was finding it annoying to swap implementation due to multimethods, but that's intentional because under your design you *should not* swap implementations
U0BKWMG5B : Right. I mean, in theory it might be good for testing, but in practice I think it makes more sense to substitute keys in the configuration, rather than make the config->implementation bridge dynamic in some fashion
U0BKWMG5B : It also makes it explicit where you're stubbing/mocking.
```