**Prerequisites**

1. Javascript & Node.js@14.16.1 (newer versions probably work too)
2. Get an IDE, we used *JetBrains WebStorm*, alternatively *Visual Studio Code*.

**Pulling and starting the Chrysalis App:**

1. Pull these Repositories:
   1. *CHRYSALIS* (main branch for the running version, webpack-upgrade branch alternatively, but that one is broken currently)
   2. *enzian-yellow* (main branch for the running version, hyperledger branch alternatively)
   3. *chrysalis-server (main)*
   4. *enzian-yellow-hyperledger-chaincode* (if you want to use the HL branches)
2. in enzian-yellow:
   1. run “npm install”
   2. , then “npm run build” (might be skippable? If not, you might have to re-run this to apply changes for linked projects)
   3. , then “npm link”
3. in chrysalis:
   1. remove enzian-yellow dependency if it exists (from package.json)
   2. run “npm link enzian-yellow” 🡪 this sym-links enzian-yellow as a dependency into chrysalis, so changes will immediately affect chrysalis without having to re-install (however dependencies aren’t propagated upwards this way!)
   3. “npm install”
4. run ‘npm install’ on chrysalis-server (and on enzian-yellow-hyperledger-chaincode if used)
5. To start the Persistence-Layer, make sure you’ve followed the steps described in the section ‘**Persistence Layer**’, then execute ‘npm run dev’ inside the *chrysalis-server* repository.
   1. Make sure that the *chrysalis-server* app port and the configured port in *chrysalis* coincide.
6. To start an Ethereum-Instance on your System, follow the steps in the section ‘**Ethereum**’.
   1. TODO connecting to it and initialization
7. Having followed these steps, you should now be able to start the *chrysalis* app by executing ‘npm run start’ at the project root of *chrysalis*. The CLI should then give you the IP/Port address, at which your browser can reach the app

**Persistence Layer:**

Please refer to the file “README.md” of the chrysalis-server repository!

**Ethereum:**

1. **Create a new Ethereum Network**: You only have to do this the first time.  
   🡪Follow the steps in <https://medium.com/swlh/how-to-set-up-a-private-ethereum-blockchain-c0e74260492c> - however some parts are outdated, refer to the points below:
   1. The genesis Block we use is provided in   
      *chrysalis-resources/configs/Ethereum/genesis.json*
   2. Our network-creation command used a different protocol, see   
      *chrysalis-resources/setup/Ethereum/setup-commands.txt*
2. **Start the Ethereum Network**: Follow the steps provided in   
   *chrysalis-resources/setup/Ethereum/start-commands*
   1. A running Node and an attached + running miner and is necessary to operate on the ETH blockchain.

**Hyperledger:**

**Warning:** Hyperledger Functionality in Enzian-Yellow is currently only under the Hyperledger-Branch. You can run “npm run TestHL” there if you deployed the chaincode right and the network is in the right state (the network setup script is fragile though), but chrysalis currently cannot integrate this branch.

1. **Setup the Hyperledger test-network** as explained here: <https://hyperledger-fabric.readthedocs.io/en/release-2.2/whatis.html>
   1. Use the “Getting Started” section for the setup.
   2. We used version 2.2 – your experience and compatibility might suffer if you use a different one.
   3. Don’t forget to use the flag “2.3.2” in the curl command to get our version.
2. Play around with the network a bit, as the tutorials also suggest (they’ll show you how Chaincode is deployed, invoked and so on)
3. **Deploy** the enzian-yellow-chaincode:
   1. Clone *enzian-yellow-hyperledger-chaincode*
   2. Follow the steps in *chrysalis-resources/setup/Hyperledger/deploy-chaincode.txt*
4. **Test-invoke** the chaincode by using the *peer chaincode invoke* command at the bottom of the *deploy-chaincode.txt* to try if you were successful. The answer (Process ID) should increment with every repetition (except when you’re doing it too fast, see the *minor issues.txt* for a bug description). Using the ‘query’ command would invoke the contract but not allow it to change the state, therefore not really deploying a process!