PLE IT 461 - Implementing Internet Applications: Overview

Module Outline

7. Blockchain Lab using EOSIO

- **7.1.Blockchain Platforms**
  - 7.1.1. Top Blockchain Platforms
  - 7.1.2. Choosing a Development Platform: EOS
- **7.2.Setting up EOSIO IDE**
  - 7.2.1. Setting up EOSIO on Gitpod
  - 7.2.2. The IDE Environment
  - 7.2.3. Create User Accounts
- **7.3.Creating a Smart Contract**
  - 7.3.1. Hello Bob
  - 7.3.2. Add Authorization to Hello Bob
- **7.4.A Smart Contract with Tokens**
  - 7.4.1. Deploy, Issue, and Transfer Tokens
  - 7.4.2. An Escrow Example
- **7.5.Next Steps**
  - 7.5.1. Final Project
  - 7.5.2. From Here

7. Blockchain Lab using EOSIO

7.A. Assignments

Assignments	Description	Deliverables	Due Date
1. Readings	Read the required reading in Resources section of this module.		Aug. 1, 2020 - 11:59 pm
2. Complete the module content			Aug. 1, 2020 - 11:59 pm
3. Participate in the module discussion forum		Discussion forum postings	Aug. 1, 2020 - 11:59 pm
4. Final Project	**Submission Guidelines** See assignment instructions on the "Final Project" page towards the end of this module.	Documentation containing screenshot of the project result	Aug. 8, 2020 - 11:59 pm
5. Submit module feedback	Please complete the feedback form for this module. Your feedback is valuable and will be used for the current and future offerings of this course.	Feedback Form	

7.1.Blockchain Platforms

7.1.1. Top Blockchain Platforms

Blockchain platforms allow the development of blockchain-based applications. Ethereum, Hyperledger, R3, Ripple, and EOS are a few names that have built blockchain frameworks, allowing people to develop and host applications on the blockchain.

Below are some factors that can help you decide the best blockchain platform for your business.

What is the adoption/maintenance status of a platform? Has the platform been widely adopted, or is the platform still in the development stage? How active is the development of a platform in GitHub or other related communities?

What type of Blockchain do you require? Do you want a public, private, or consortium (permissoned) blockchain network? Do you need a digital currency for your application?

What Programming Languages does the platform support? Blockchain is still in its nascent stage and many new programming languages have been introduced. Before selecting a blockchain framework, know what programming languages are supported by the SDKs of that platform, and whether you have experience with these programming languages.

Who owns the platform and how popular is it? Look at the reputation of a blockchain platform on websites like GitHub or Reddit.

What Blockchain Consensus Protocol does the platform use to achieve consensus? It is challenging to work on the blockchain platform if you don't understand its consensus protocol. Since there are different types of consensus algorithms such as Proof of Work, Proof of Stake, Proof of Burn and many more, it is crucial to understand that which blockchain platform supports what consensus protocol.

Does the platform support Smart Contracts Functionality? Not all blockchain platforms support the concept of smart contracts which are responsible for validating, triggering or enforcing actions on the blockchain systems.

What kind of scalability does your solution need? As the number of transactions and participants increase, a blockchain network should be able to scale to adapt to the growth. How many transactions per second can be performed on the blockchain platform? What is the transaction fee on average? You need to consider quality, ease of use, access, security, speed and cost.

Below is a comparison of 13 top blockchain platforms in 2020, reviewed by LeewayHertz, a application development company:

	Ethereum	Hyperledger Fabric	R3 Corda	Ripple
Industry focus	Cross-Industry	Cross-Industry	Financial Services	Financial Services
Ledger Type	Permissionless	Permissioned	Permissioned	Permissioned
Consensus Algorithm	Proof of Work	Pluggable Framework	Pluggable Framework	Probabilistic Voting
Smart Contract	Yes	Yes	Yes	No
Governance	Ethereum Developers	Linux Foundation	R3 Consortium	Ripple Labs

7.1.2. Choosing a Development Platform: EOS

Out of the potentially many blockchain platforms/technologies that satisfy a business need, everyone wants to choose a technology platform that offers the brightest future and, yet, meets the time and cost constraints. This course module has choosen EOS as the development platform. Given the rapid advancements in this field, it will not be surprising if another blockchain platform becomes a better choice in the future. You are welcome to contact the instructor if the information on this page is out of date or inaccurate.

One way to evaluate the success of a blockchain platform is through its market capitalization. Another way, closer to the real value, is observing the activity on the blockchain. Lots of hype is responsible for "misplaced" capital in the cryptocurrency space. Which project is actually being used by people is one of the best predictive indicators of the value of the blockchain platform/technology.

Blocktivity, blocktivity.info, ranks popular blockchain platforms by the amount of activity, i.e., the number of transactions carried out on the platform. At the Blocktivity website, visitors can also compare the market cap and other volume indicators of the blockchains. As of May 2020, EOS is ranked at the top by activity (https://blocktivity.info), and is a major platform by market cap.

One objective of this module is to use a fast and cheap platform to implement smart contracts on blockchain. Such a platform has high potential to be adopted by enterprises for high-volume transactions. Ethereum is the first blockchain that has provided smart contracts functionality, and it remains popular among blockchain developers. However, EOS has significant performance advantages over Ethereum for learning brand new application development:

Transaction time: The time that it takes for an Ethereum transaction to complete depends on how much you pay in gas (i.e., transaction fees) and how congested the Ethereum network is. On an average day in 2019, it took anywhere between 15 seconds and 5 minutes to process a transaction, if you paid the standard gas price (https://ethgasstation.info/blog/ethereum-transaction-how-long/). On EOSIO, transactions complete under a second.

Transaction fees: The average fee per transaction on Ethereum is $0.33 as of May 16, 2020 (ycharts.com). Transactions are free on EOSIO. EOSIO eliminates the transaction fees through inflation, rather than by making the individual users of the network pay to access its combined resources. In other words, ownership of tokens covers any incidental costs involved.

Of course, Ethereum and other platforms are constantly improving their technologies, in order to be competitive. It remains to be seen which blockchain technology will become dominant for enterprise applications in the coming years. Please read Blockgeeks' article "What is EOS Blockchain: Beginners Guide" at: https://blockgeeks.com/guides/eos-blockchain/. Below is the video from the article:

Embedded Content - Available Online Only
Source: https://www.youtube.com/watch?v=46XOIH5wLYo.

7.2.Setting up EOSIO IDE

7.2.1. Setting up EOSIO on Gitpod

EOS, by block.one (the name of company maintaining the EOS platform), is the blockchain network that functions like an operating system: supplying the core functions including databases, accounts with permissions, scheduling, authentication, and communication, thus allowing developers to focus on their own business process. EOS is also the name of the indigenous cryptocurrency on this blockchain network.

EOSIO is the software/architecture that allows companies to construct blockchain applications that resemble an Internet software application.

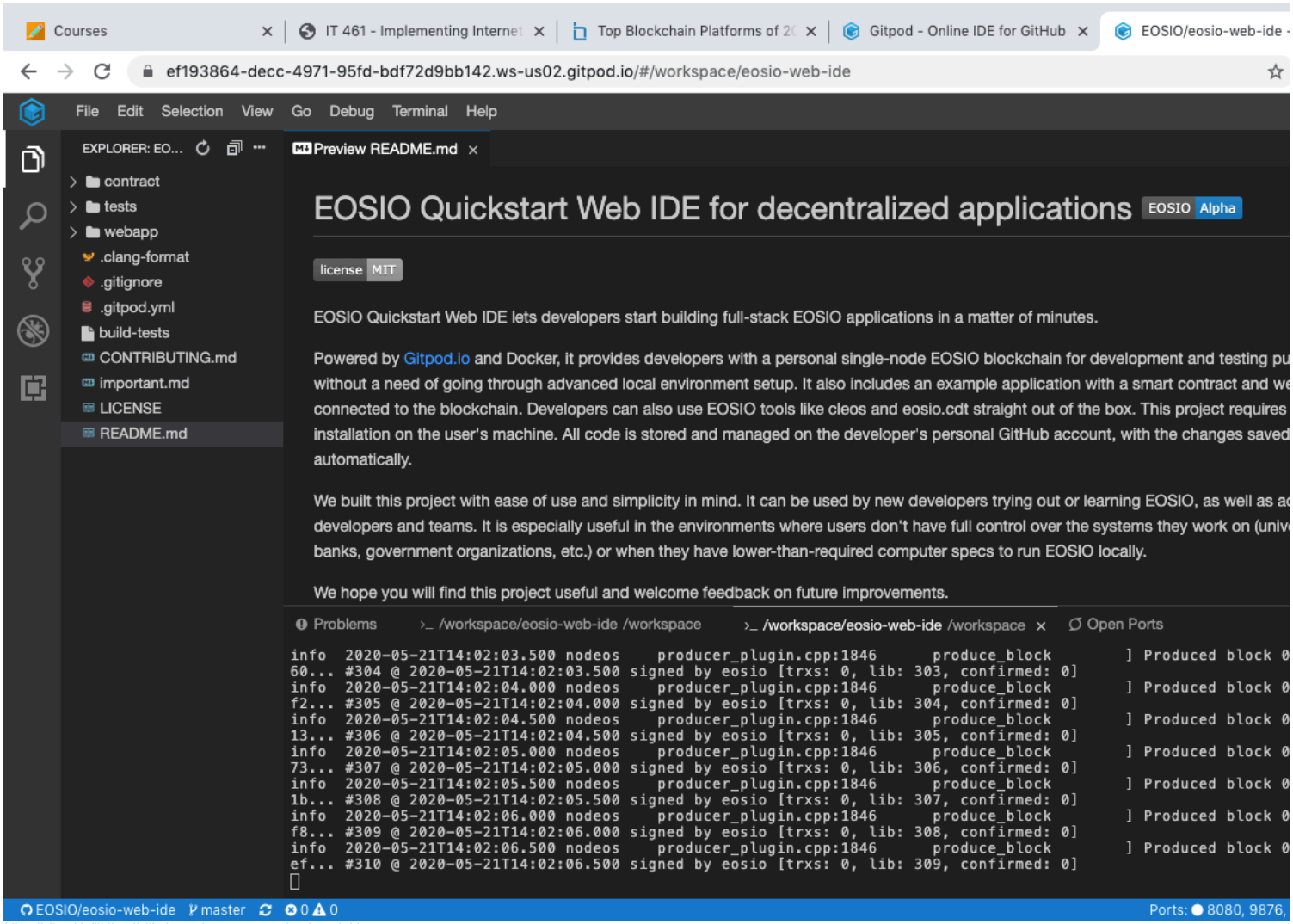
For beginners, the easiest way to learn EOSIO development is to use a virtual environment. Please follow the steps below to set up an EOSIO environment on Gitpod (register for Github and Gitpod if you do not yet have an account with them).

- Make a fork of https://github.com/EOSIO/eosio-web-ide into your own Github account

- Start your EOSIO IDE (Integrated Development Environment) by going to https://gitpod.io/#https://github.com/yourusername/eosio-web-ide (replace yourusername with your actual github username)

- Note that you may have created multiple workspaces (virtual environments) when you open and close the browser. You can check your workspaces at https://gitpod.io/workspaces/, and stop or start the workspace if you choose. When you restart/reinitialize a workspace, you may need to recreate the test users and re-run the steps in this module because the objects on your personal blockchain are lost when the workspace is reinitialized.

Below is what you will see when you start your Gitpod IDE. One of the tabs at the bottom of the screen shows the the continuous creation of blocks (please do not close this tab). The blocks will carry the transactions resulting from this lab module, as explained in the previous module. On a production EOS blockchain, the blocks will be broadcast to all participants.



https://gitpod.io/#https://github.com/yourusername/eosio-web-ide

---

7.2.2. The IDE Environment

The Gitpod IDE greatly simplifies the environment setup by seeding the environment with default settings. While you enjoy the convenience of this setup-free environment, you should understand what has been done in the background. You should also learn using Linux terminal, a command line interface.

The Gitpod IDE sets up and initalizes an environment using the code repository and settings stored at https://github.com/your-github-account/eosio-web-ide, which is a fork of https://github.com/EOSIO/eosio-web-ide. The IDE is installed on the default docker (virtual machine) image of Gitpod, which is Ubuntu.

The Gitpod IDE is equivalent to an environment set up using the Development Environment instructions on the EOSIO Developer Portal (Sections 1.1 through 1.5).

https://developers.eos.io/welcome/latest/getting-started/index

You should read through Sections 1.1 through 1.5 of the above Getting Started Guide to understand the environment setup procedure (required if you were to install EOSIO IDE on a local computer). However, please do not run any setup steps on your Gitpod virtual machine because those steps (or equivalent) have already been run when you start up the forked/cloned virtual machine. Please pay special attention to Section 1.4, Create Development Wallet: every time you access Gitpod IDE, the initialization process creates, opens, and unlocks the default *wallet*, created/stored one key (pair) in wallet, and creates the default system user *eosio*. The public key of this pre-created key pair stored in the wallet will be used for various commands in this module.

To get familar with the Gitpod IDE, and to view the pre-created wallet and key, you should run the following exercise:

Open a terminal using the Terminal drop-down menu in the IDE user interface, and issue following commands:

lsb_release -a

The result should show the Linux version of the Gitpod docker.

cleos wallet list

The result should show the following:

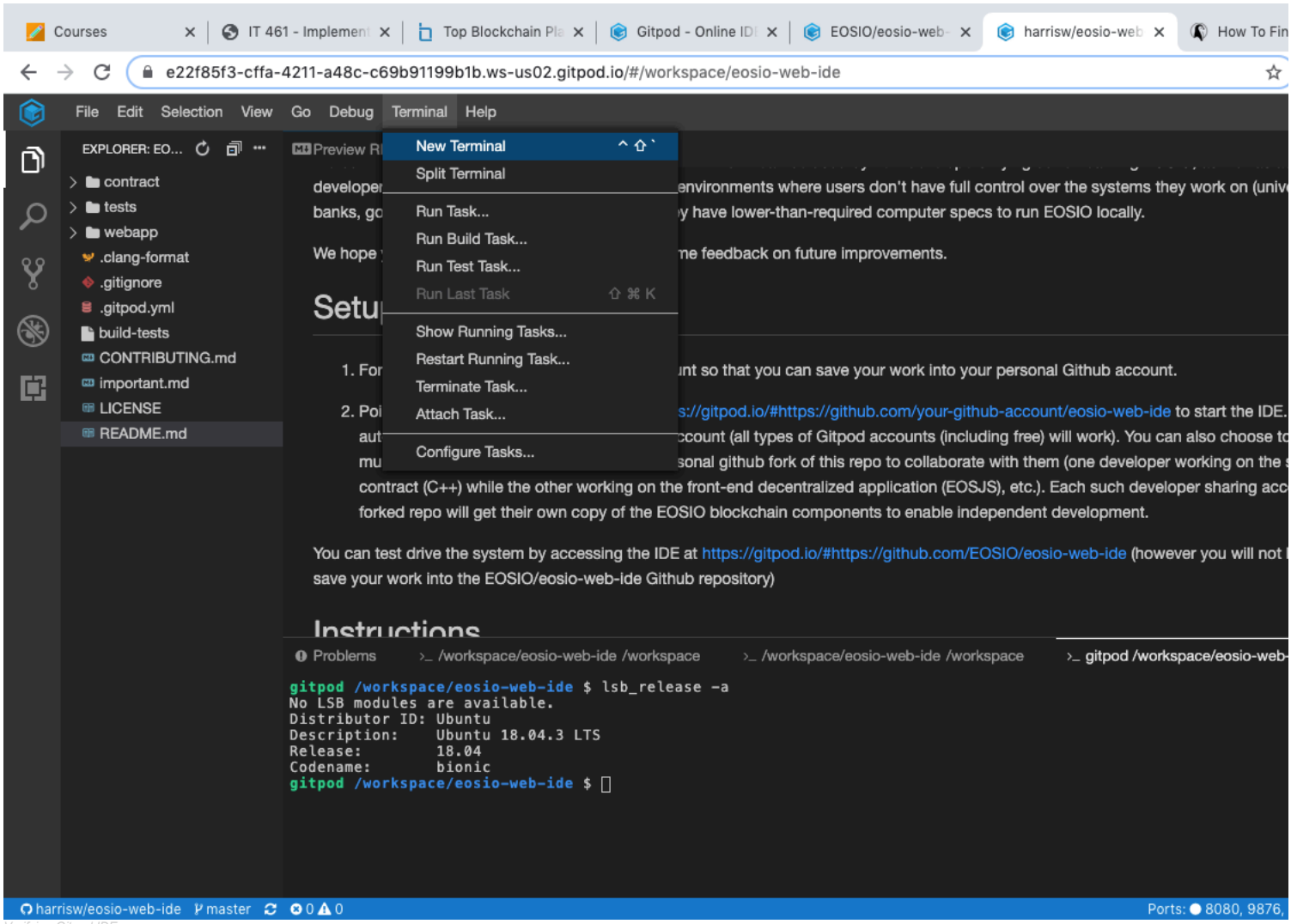
```
Wallets:
[
  "default"
]
```

cleos wallet keys

The results should show the public key created during the Gitpod IDE initialization process.

Optional: Once you have this public key, you can go to Section 1.4, the Create Development Wallet page on the EOSIO develpment portal, and enter the key into a textbox on this page. The rest of the tutorial pages on the EOSIO development portal will be populated with this public key to make it easier for you to copy the commands from the tutorial.

cleos get account eosio
The result should show some information related to the user, indicating that the user has been created by the Gitpod initialization process.



Verifying Gitpod IDE

7.2.3. Create User Accounts

An account is a collection of authorizations, stored on the blockchain and used to identify a sender/recipient. It has a flexible authorization structure that enables it to be owned either by an individual or by a group of individuals, depending on how permissions have been configured. An account is required to send or to receive a valid transaction to the blockchain.

This lab module uses two "user" accounts, bob and alice, as well as the default eosio account already created when the Gitpod IDE environment was started/initialized. Please follow the instructions in Section 1.6, Create Test Accounts, on EOS Developer Portal:

https://developers.eos.io/welcome/latest/getting-started/development-environment/create-test-accounts

You will need to specify a valid public key when creating a user account. For now, we can just use the existing key in the default wallet created when the Gitpod environment is

started/initialized (i.e., the output of the "cleos wallet keys" command on your Gitpod environment). In this module, we will push various actions to the blockchain using different user accounts on the Gitpod virtual machine, to simulate the real world situation where multiple users interact with the blockchain using different accounts from different computers.

Below are the commands to create the user accounts as the system user *eosio*, using the public key string with the key existing in your wallet.

cleos wallet keys

```
[
  "EOS6MRyAjQq8ud7hVNYcfnVPJqcVpscN5So8BhtHuGYqET5GDW5CV"
]
```

cleos create account eosio bob a*-public-key-in-your-wallet*

cleos create account eosio alice a*-public-key-in-your-wallet*

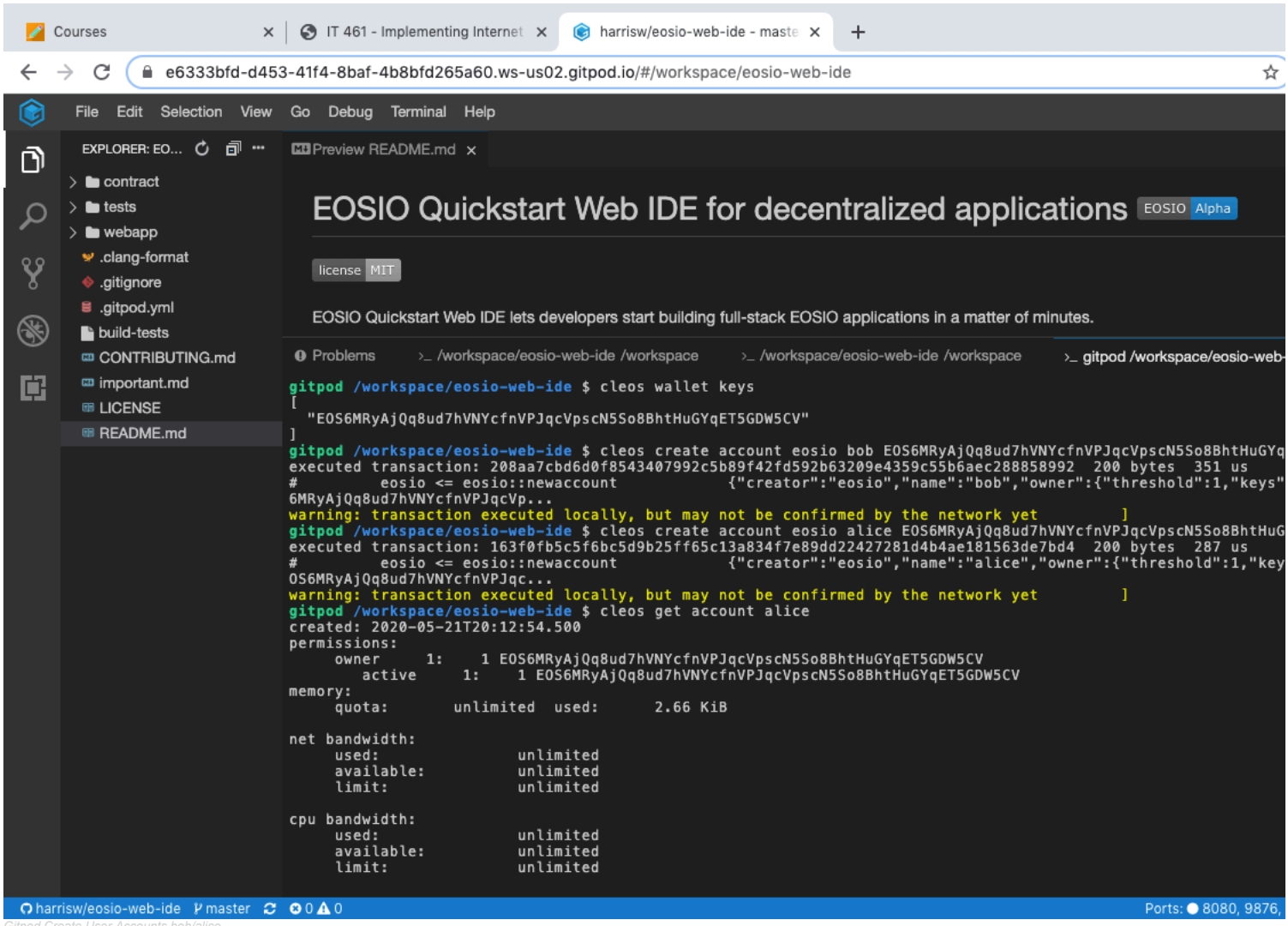
Be aware that the account name is the only identifier for ownership. You can change the public key, but it would not change the ownership of your EOSIO account.

Verify the account information of alice, including which public key is associated with the alice account:

cleos get account alice

When you restart/reinitialize a workspace, you may need to recreate the test users and re-run the steps in this module because the objects on your personal blockchain are lost when the workspace is reinitialized.

Below is the screenshot of the commands and output on Gitpod.



Gitpod Create User Accounts bob/alice

7.3.Creating a Smart Contract

7.3.1. Hello Bob

A smart contract on a blockchain can be used to develop many different types of applications. Here we start with a simple "hello bob" messaging task.

For the steps below, please refer to Section 2.1 of the Getting Started tutorial on EOSIO Developer Portal, at: https://developers.eos.io/welcome/v2.0/getting-started/smart-contract-development/hello-world

- Create a new directory called "hello" in the contracts directory and enter the directory.

cd /workspace/eosio-web-ide/contract
mkdir hello
cd hello

- Create a new file, "hello.cpp" and open it in your favorite editor.

touch hello.cpp

You can use a command line editor, such as vim. Or, you can just click on the hello.cpp in the file/directory tree and use the code editor with the webpage on your browser. Depending on whether you have installed the browser extension for Gitpod, and whether your customized the layout of various frames within the webpage, the position and size of the code editor frame may be different.

- Enter the following as the content of hello.cpp. If you use the built-in code editor on the webpage, make sure that you save the file, using the File menu.

```
#include <eosio/eosio.hpp>
using namespace eosio;
class [[eosio::contract]] hello : public contract {
  public:
     using contract::contract;
     [[eosio::action]]
     void hi( name user ) {
       print( "Hello, ", user);
     }
};
```

- Compile the code

eosio-cpp hello.cpp -o hello.wasm

- Using the system user *eosio*, create a new account *hello* to deploy the code:

cleos create account eosio hello EOS6MRyAjQq8ud7hVNYcfnVPJqcVpscN5So8BhtHuGYqET5GDW5CV -p eosio@active
cleos set contract hello /workspace/eosio-web-ide/contract/hello -p hello@active

Note that the option "-p xxxx@yyyy" indicates the permission level used by the cleos action. In other words, the cleos action is taken on behalf of this given account xxxx on a certain permisison level yyyy. You can find all of the options and their meanings using the -help option:

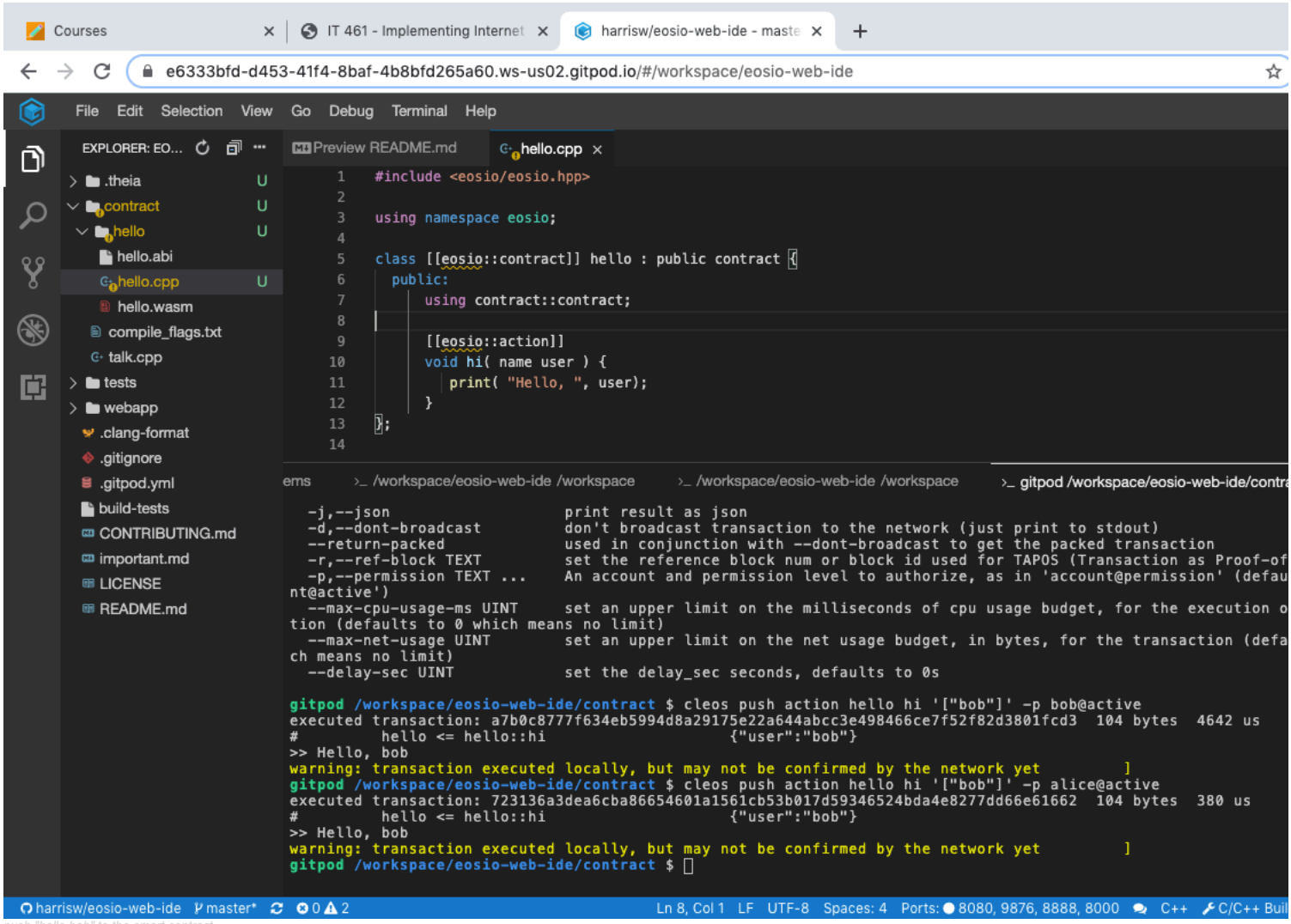
cleos -help
cleos create account -help
cleos set contract -help

```
    -p,--permission TEXT ...    An account and permission level to authorize, as in 'account@permission' (defaults to 'creator@active')
```

- Now the contract is set, push an action to the contract.

cleos push action hello hi '["bob"]' -p bob@active
cleos push action hello hi '["bob"]' -p alice@active

As you can tell, anyone (either bob or alice, without any special privileges) can push an action to the contract and print the message "hello bob".



push "hello bob" to the smart contract

7.3.2. Add Authorization to Hello Bob

To add some authorization to the *print* action, add one single line to *hello.cpp*. The *require_auth* method takes a name as a parameter, and will check if the user executing the action matches the provided parameter. With this authorization requirement, only the user can say hello to herself. This simple authorization is for the purpose of illustration; a real-world authorization can be based on more complex business rules.

```
void hi( name user ) {
  require_auth( user );
  print( "Hello, ", name{user} );
}
```

After editing hello.cpp (again, make sure you save the edit), re-compile the code and deploy the contract:

eosio-cpp -o hello.wasm hello.cpp
cleos set contract hello /workspace/eosio-web-ide/contract/hello -p hello@active

Now, try to push a hello "bob" action to the smart contract using *alice* account:

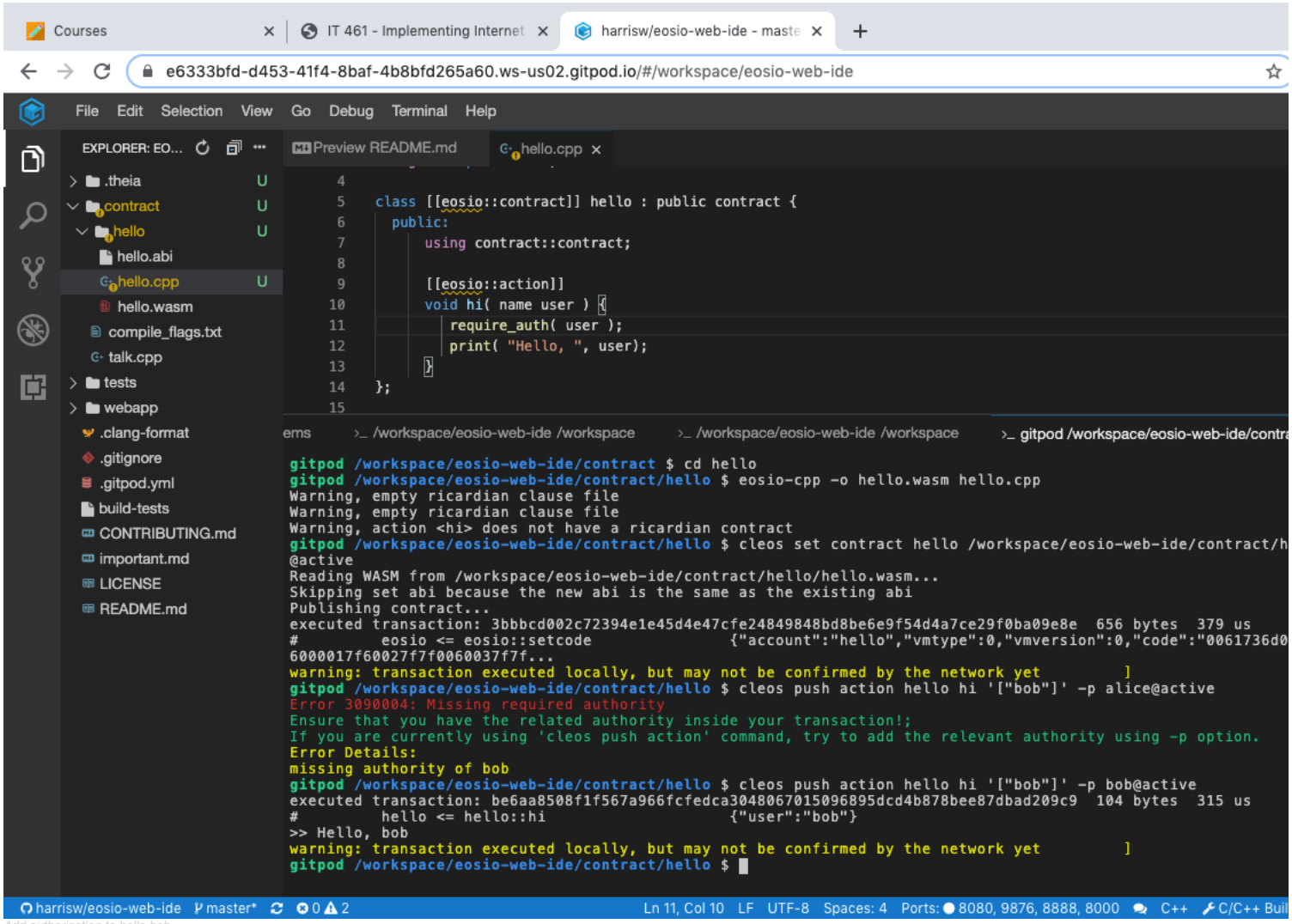
cleos push action hello hi '["bob"]' -p alice@active

The authorization fails with the following error messages because the account pushing the message is alice, and "alice" does not match the name "bob".

Ensure that you have the related authority inside your transaction!;
If you are currently using 'cleos push action' command, try to add the relevant authority using -p option.
Error Details:
missing authority of bob

Try push a hello "bob" action to the smart contract using the *bob* account. It should successful print "hello bob".

cleos push action hello hi '["bob"]' -p bob@active



Add authorization to hello bob

7.4.A Smart Contract with Tokens

7.4.1. Deploy, Issue, and Transfer Tokens

Blockchain was orginally designed to record transactions of digital assets in a distributed ledger. The digital assets of value are often called tokens, coins, or cryptocurrencies.

To work with tokens, we build on a pre-defined smart contract, eosio.token, from the EOSIO code library. The instructions below are adopted from Section 2.2, Deploy, Issue, and Transfer Tokens, of the Getting Started tutorial on the EOSIO Developer Portal.

https://developers.eos.io/welcome/latest/getting-started/smart-contract-development/deploy-issue-and-transfer-tokens

First, copy the eosio.token from the EOSIO code library and compile the code. Then create a user account, and deploy the token smart contract under this newly created user account.

cd /workspace/eosio-web-ide/contract
git clone https://github.com/EOSIO/eosio.contracts
cd eosio.contracts/contracts/eosio.token

cleos create account eosio eosio.token EOS6MRyAjQq8ud7hVNYcfnVPJqcVpscN5So8BhtHuGYqET5GDW5CV
eosio-cpp -I include -o eosio.token.wasm src/eosio.token.cpp --abigen
cleos set contract eosio.token /workspace/eosio-web-ide/contract/eosio.contracts/contracts/eosio.token --abi eosio.token.abi -p eosio.token@active

Note that the "-abigen" option is specified in the compile command eosio-cpp. The -abigen option specifies to generate ABI, the Application Binary Interface, a JSON-based description on how to convert user actions between their JSON and Binary representations. The generated ABI file can be viewed under the same folder with the source code. While not used in our programming, the ABI file can be used to inspect the interface provided by the smart contract for user actions. For other compile options, you may use "*eosio-cpp -help*".

Now that the eosio.token application is deployed under the eosio.token user account, the smart contract owner eosio.token user can create/define a token/currency and designate a token issuer. This issuer will have the authority to call *issue* and other token actions such as closing accounts or retiring tokens. In this case, let alice be the token issuer. The token definition contains two pieces of data, a floating-point number setting the maximum supply and a symbol in capitalized alpha characters representing the asset.

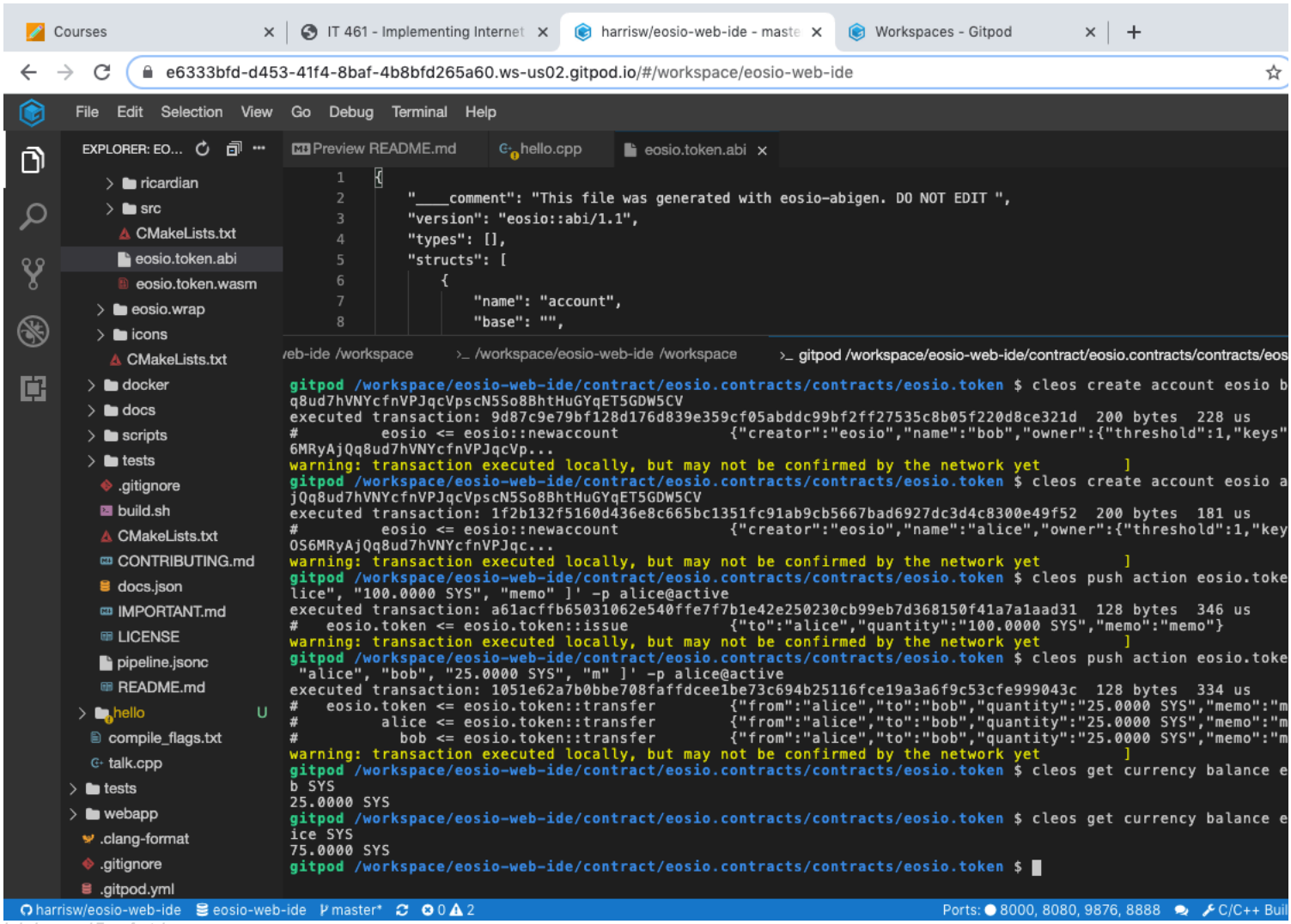
cleos push action eosio.token create '{"issuer":"alice", "maximum_supply":"1000000000.0000 SYS"}' -p eosio.token@active

Now that the token SYS is defined, the issuer alice can issue new tokens. As mentioned earlier only the issuer can do so, therefore, -p alice@active must be provided to authorize the issue action.

cleos push action eosio.token issue '["alice", "100.0000 SYS", "memo"]' -p alice@active

Now that the issuer alice has 100.0000 tokens, alice can transfer some tokens to account bob.

cleos push action eosio.token transfer '["alice", "bob", "25.0000 SYS", "m"]' -p alice@active



Lab: Issue and Transfer tokens

7.4.2. An Escrow Example

Here, we will explain how to write a smart contract that has an escrow, or hold, action. We are going to write a contract that will accept SYS tokens but will not allow the deposited tokens to be withdrawn until a certain time. For complete source code and more detailed steps, please refer to Section 2.9: Payable Actions in the Getting Started tutorial on EOSIO Developer Portal: https://developers.eos.io/welcome/latest/getting-started/smart-contract-development/payable_actions

First, create a standard class called "hodl" that extends eosio::contract. Note that "hodl" is a slang in the cryptocurrency community for holding the cryptocurrency rather than for selling

it. It orginated from the Bitcoin culture, and it describes a stance of "to stay invested in bitcoin and not to capitulate in the face of plunging prices." The usage of "hodl" has gradually grown into the investment world. In this exercise, we simply use "hodl" as the name of the C++ class to implement the holding action.

The requirement of this simple escrow contract is that users must make token deposits before a certain deadline. Then, the rightful owner "xxx" can withdraw the money only after this deadline, defined by the constant "the_party" in the escrow contract, by pushing an action, *party "xxx"*, to the smart contract on the blockchain.

For the deposit action, there are following checks and the associated error messages:

check(now() < the_party, "You're way late");
check(quantity.amount > 0, "When pigs fly");
check(quantity.symbol == hodl_symbol, "These are not the droids you are looking for.");

For the withdraw action, also called the "party" action, there are following checks:

check(now() > the_party, "Hold your horses");
check(hodl_it != balance.end(), "You're not allowed to party");

Only if the current time now() > the_party, and when there is sufficient balance, the "party" action transfers back the tokens in the escrow account to the rightful owner. Please refer to a C++ reference manual if you have trouble understanding the syntax in above statements in italics.

To create this hodl class, you can create a hodl directory and edit the hodl.cpp:

cd /workspace/eosio-web-ide/contract
mkdir hodl
cd hodl
touch hodl.cpp

You can copy into hodl.cpp the complete code listing from https://developers.eos.io/welcome/latest/getting-started/smart-contract-development/payable_actions. In the code listing, the_party time is originally defined as a future time, which can be left as is. Then, compile *hodl.cpp* and deply the *hodl* smart contract under a new user account, *hodl*. This hodl smart contract needs to make inline action calls and, therefore, need an additional permission "--add-code".

eosio-cpp hodl.cpp -o hodl.wasm
cleos create account eosio hodl EOS6MRyAjQq8ud7hVNYcfnVPJqcVpscN5So8BhtHuGYqET5GDW5CV
cleos set contract hodl ./ -p hodl@active
cleos set account permission hodl active --add-code

The following commands assume that you have deployed the EOS token in the previous section, and that alice has a non-zero token balance. If you have reintialized the workplace and lost the test users, you need to recreate the *alice* user and *eosio.token* user, and re-deploy the *eosio.token* contract and re-issue tokens to *alice*.

cleos transfer alice hodl '0.0001 SYS' 'Hodl!' -p alice@active

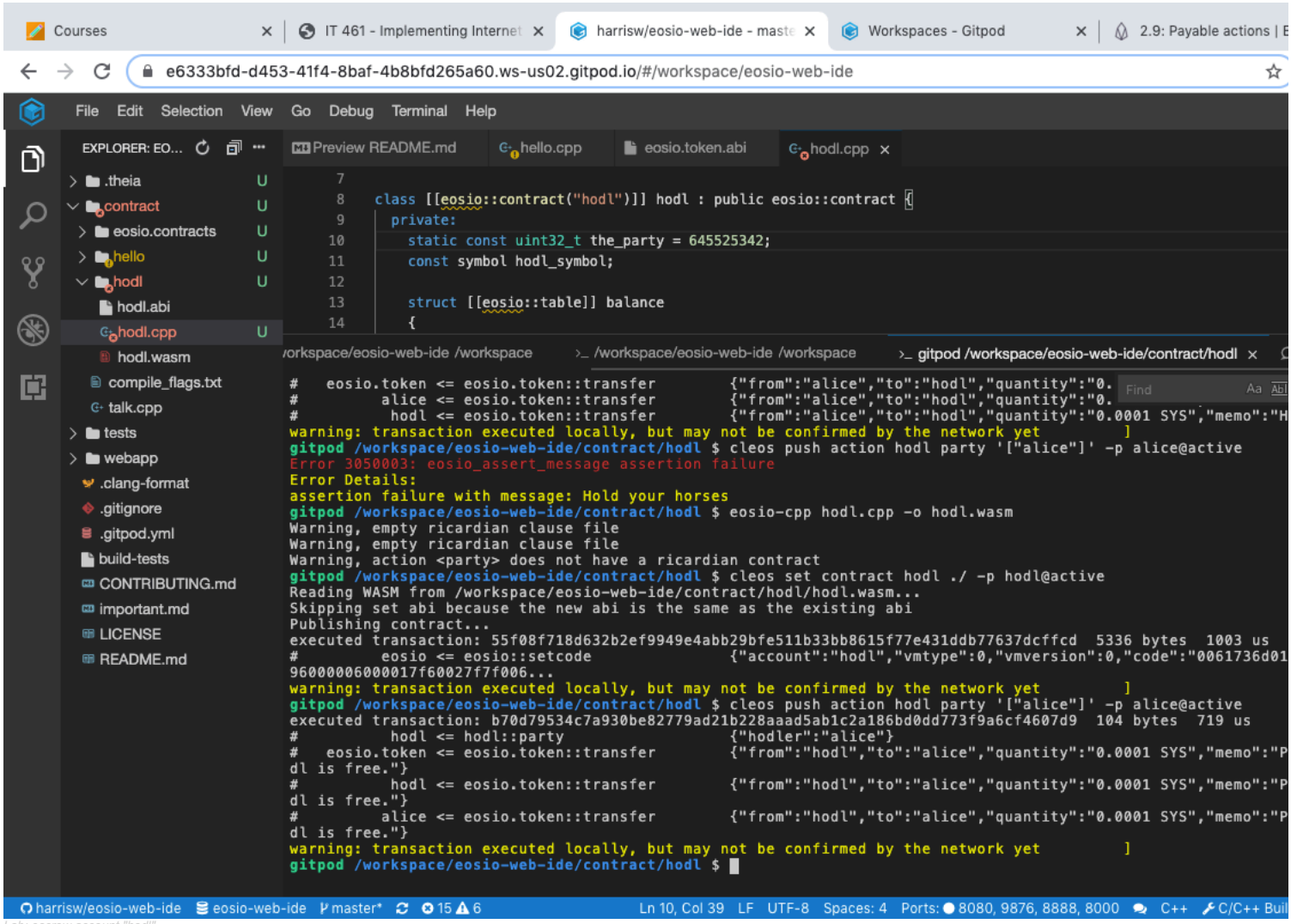
If you attempt to withdraw the money now by pushing the "party alice" message, you should receive a failure message saying "hold your horses", because the time defined by the_party in hodl.cpp is in the future.

cleos push action hodl party '["alice"]' -p alice@active

To allow withdraw, you can edit the the_party constant to be a time in the past. You can simply remove the leading digit 1 from the constant to make the constant much smaller. That will make the time constant to be in the past. Then, you need to recompile and redeploy the escrow contract:

eosio-cpp hodl.cpp -o hodl.wasm
cleos set contract hodl ./ -p hodl@active

Below screenshot shows the sample output from above commands, and also the value of the_time constant after the leading digit 1 is removed, to allow the withdraw.



Lab: escrow account "hodl"

7.5.Next Steps

7.5.1. Final Project

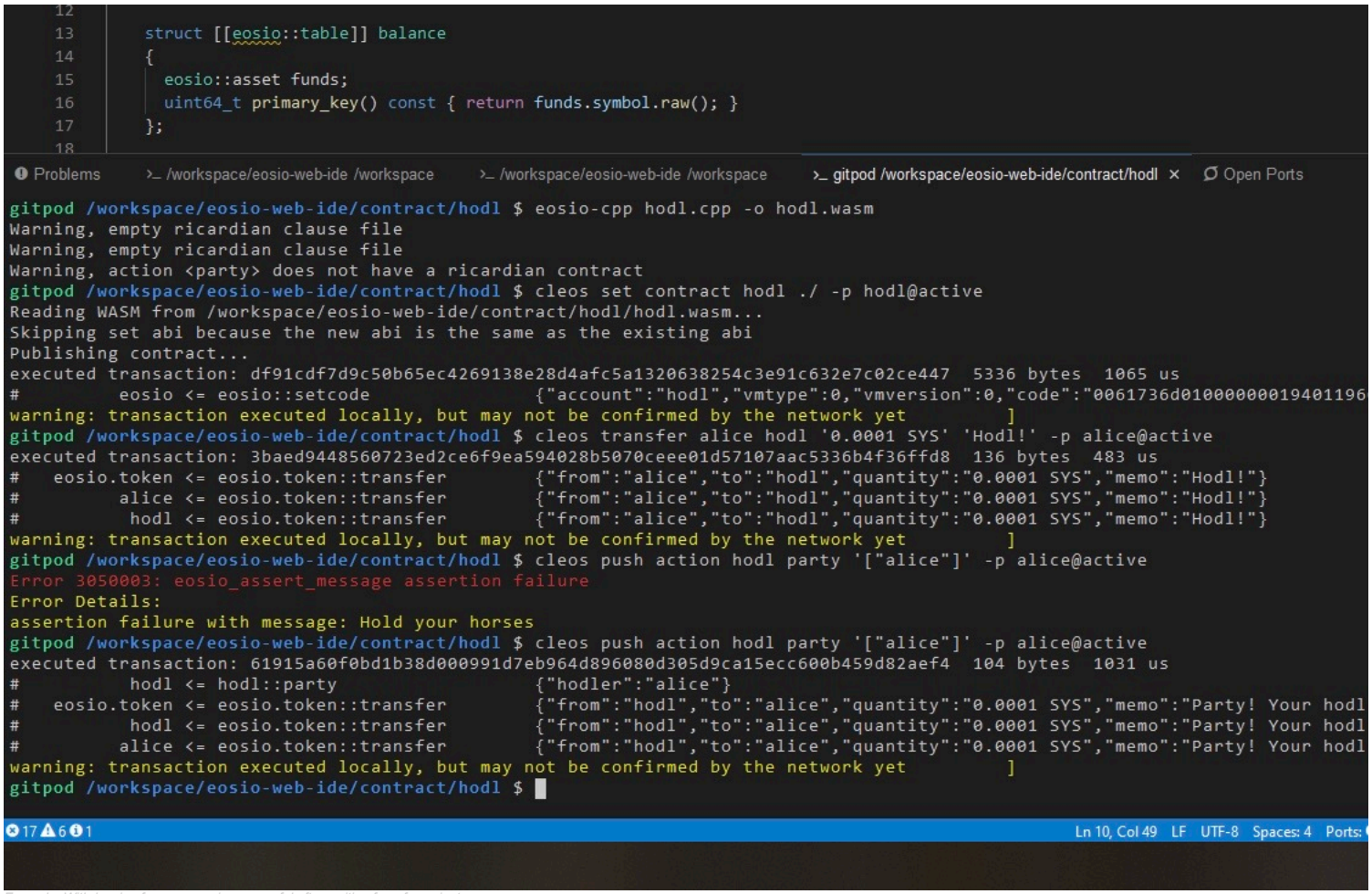
Using a Gitpod virtual machine has greatly simplied the environment setup but has its limitations. The virtual environment is transient. You need to re-run certain steps when the Gitpod environment restarts. The Gitpod virtual environment also limits the complexity of our exercises in this module.

For the final project, please modify the escrow account program (in C++ programming language) so that once the modified program is deployed, you can deposit the escrow and then receive refunds without modifying and redeploying the program in between. You have two ways to do so, with the 2nd option being open-ended and far more challenging/interesting.

1. A simple way is to set the deadline value to be an upcoming point-in-time (such as NOW + 2 minutes) so that the first time you deposit the cybercurrency, the time is before the deadline. Then if you wait for a couple of minutes, you can withdraw the cyber deposit. Please do your own research on how to set this deadline value. *The screenshot at the bottom of this page illustrates sample results from this approach.*

2. Design a different condition so that you can withdraw after making a deposit. Blockchain and smart contract are often used to make payments when a preprogrammed condition of a contractual agreement is triggered.

In either case, please take a screenshot of your complete GitPod page (containing the URL of your gitpod) showing that you can withdraw money after the deposit, without re-compiling and re-deloying the program on the blockchain. Please submit the documentation explaining your implementation, including the aforementioned screenshot of GitPod results. You are welcome to use online resources (reference manuals, Google search, etc.) if you have any questions related to C++ programming language.



Example: Withdrawing from escrow is successful after waiting for a few minutes

---

7.5.2. From Here

To continue learning EOSIO development, you are recommended to do the following:

A. Follow the tutorials at https://developers.eos.io/welcome/latest/tutorials/index:

1. BIOS Boot Sequence: Demonstrates how to bootstrap a new EOSIO blockchain.
2. Tic Tac Toe Game Contract: Build a sample Tic Tac Toe game on EOSIO.
3. Elemental Battles: Created for developers new to blockchain, Elemental Battles is part-tutorial, part-game development. No prior blockchain programming is required. You'll play in a fantasy world and you'll harness the powers of wood, water, and fire in order to learn blockchain basics and to build your first game on EOSIO. https://battles.eos.io/

B. Install a local EOSIO development environment on one of the following operating systems supported by EOSIO. You can follow the instructions at https://developers.eos.io/welcome/latest/getting-started/development-environment/introduction

Amazon Linux 2
Centos 7
Ubuntu 16.04
Ubuntu 18.04
MacOS 10.14 (Mojave) and higher

7.R. Resources

REFERENCES

No.	Title	Type	Size
1	How long does an Ethereum transaction really take? June 5, 2019. *https://ethgasstation.info/blog/ethereum-transaction-how-long/*		
2	EOS vs Ethereum – What's the Better Alternative? Updated Jan 17, 2020. *https://www.bitdegree.org/tutorials/eos-vs-ethereum/*		
3	Top Blockchains of 2020. *https://www.leewayhertz.com/blockchain-platforms-for-top-blockchain-companies/*		
4	EOS Developer Portal. *https://developers.eos.io/*		
5	EOS Technical White Paper. *https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md*		

REQUIRED READINGS

No.	Title	Type	Size
6	What is EOS Blockchain: Beginners Guide. *https://blockgeeks.com/guides/eos-blockchain/*		