DECENTRALISED APPLICATIONS

ON

TURTLENETWORK

POWERED BY TURTLENETWORK

REVISION HISTORY

Revision History		
Revision	Date	Description
1	12/4/2020	First revision
2	12/7/2020	Some corrections

CONTENTS

ACKNOWLEGEMENT

First and foremost, I would like to sincerely thank TurtleNetwork team for their wonderful work that brought TurtleNetwork into existence a few years ago, and most importantly their positive encouragement and a warm spirit to help us in almost every thing related to the blockchain technology.

I also want to extend my thanks to Polarity team for their valuable contribution in my humble guide, and for their effort to bridge the the gap between the blockchain technology and the real world usage.

Last but not least, I would like also to thank our wonderful TurtleNetwork community for their continuous support.

WHAT IS A DAPP?

DApp refers to a decentralized application or distributed application. Basically, it looks like your application downloaded from App Store or Google Play, except it is a decentralized entity built on top of a particular blockchain or any other peer-to-peer (P2P) network. Decentralization is the core concept of the blockchain industry. In the dApp context, it means that no one in the entire world can manage or control a dApp.

Traditional application
Frontend (the visual appearance of an application) – API – centralized database

Decentralized application
Frontend (the visual appearance of an application) – smart contract – blockchain

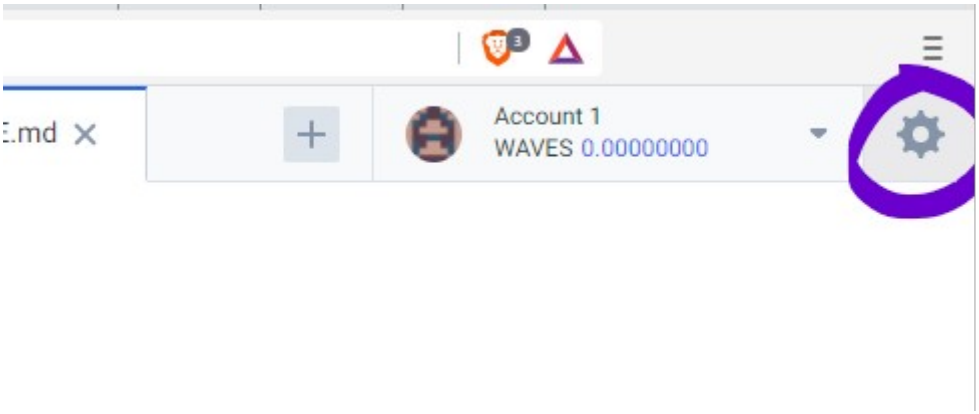
LANGUAGE FOR DAPPS ON TURTLENETWORK

To develop Dapps on the TurtleNetwork blockchain, we use RIDE langage. RIDE is a blockchain scripting language, which enables 'smart' blockchain transactions. The execution result is predicated on certain logic, realised using RIDE scripts and deployed on the blockchain. The goal of RIDE's architecture is to create a native on-chain computation layer which is as close to the general blockchain architecture (full data synchronisation) as possible.

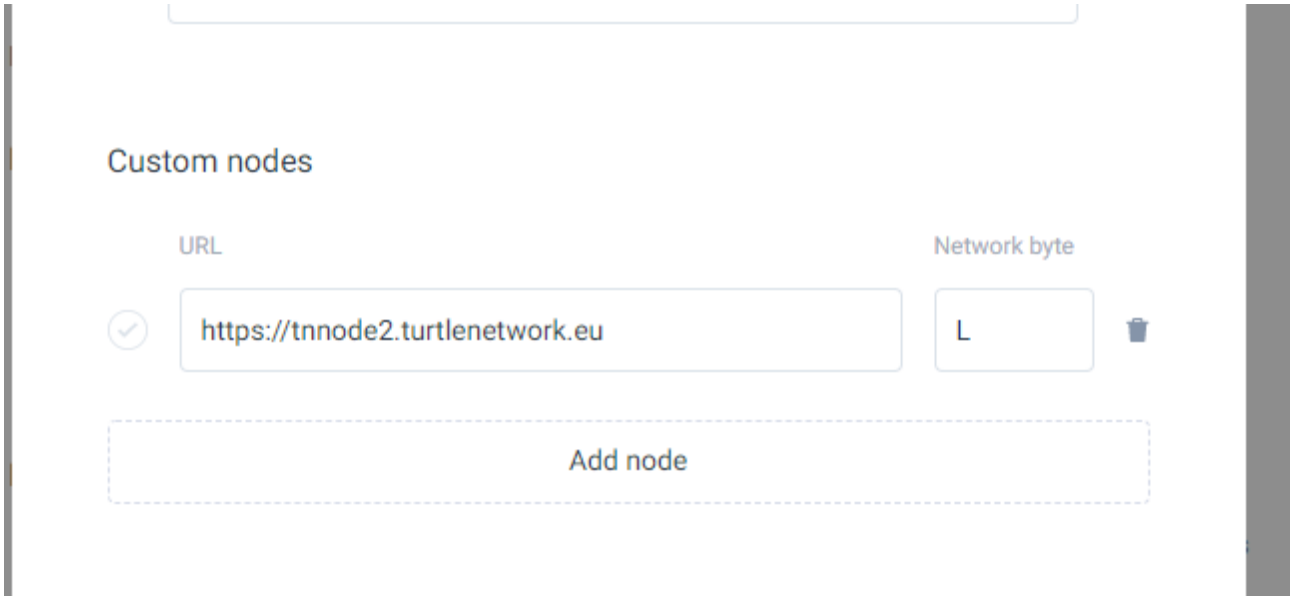
HOW TO WRITE AND PUBLISH A DAPP ON TOP OF THE TURTLENETWORK

To write and publish your first dapp on Turtlenetwork blockchain, use the following steps :

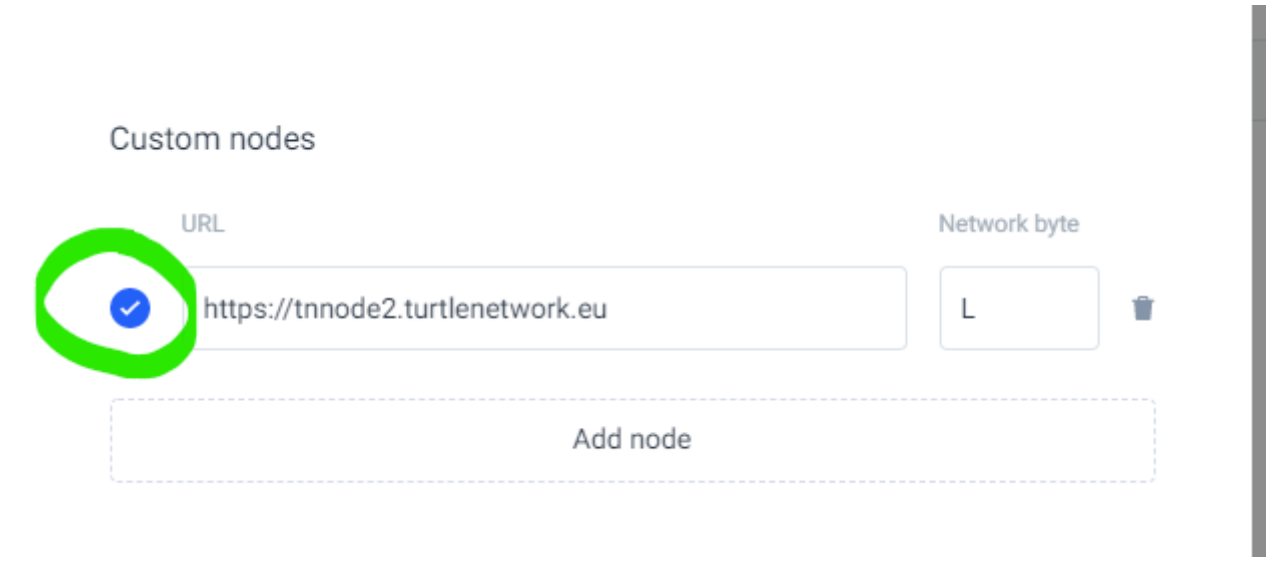
- Open the online waves IDE: https://waves-ide.com/
- Click on the setting button as shown in the pic bellow :



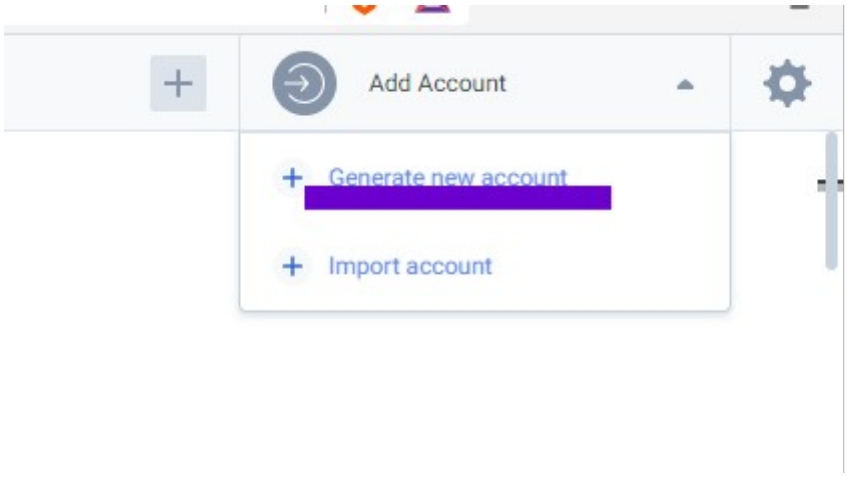
- Enter the TurtleNetwork custom node as shown on the pic below :
URL : https://tnnode2.turtlenetwork.eu
Network byte : L

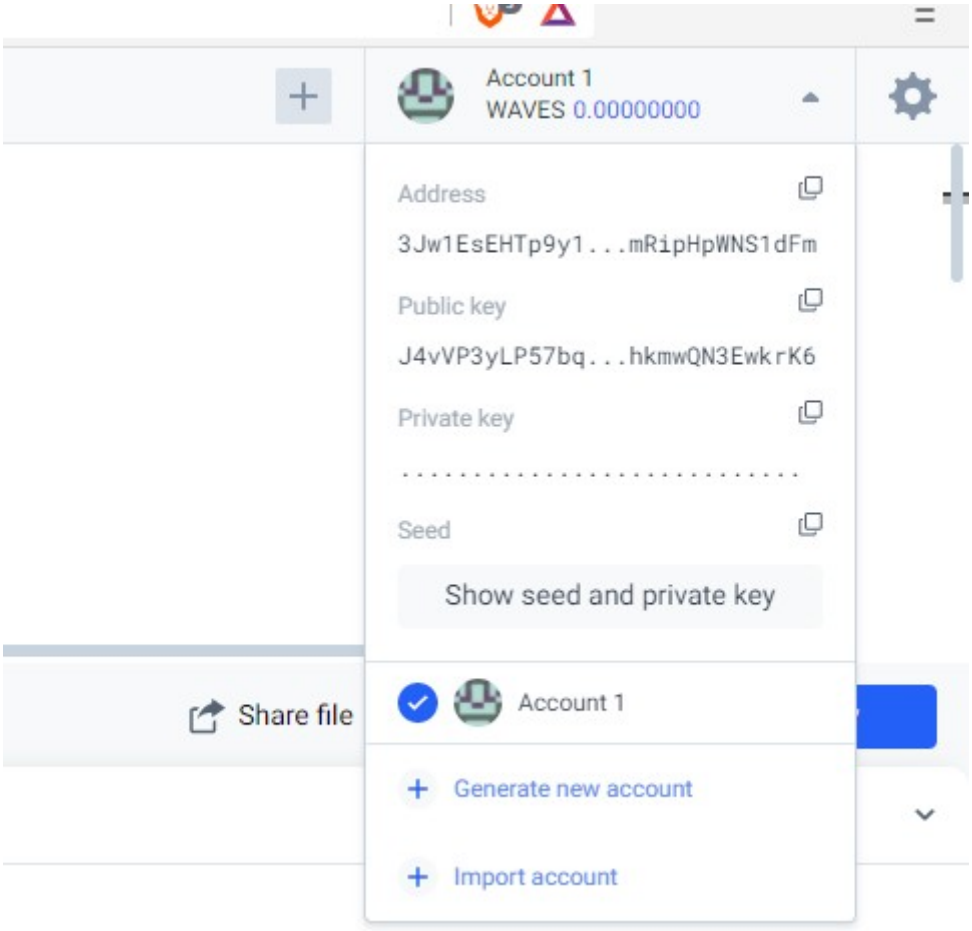


- Then activate it as your default custom node as shown below :



- After setting up the custom node, you will have to generate a new Turtlenetwork account as shown below and fill it with some TN (minimum 1 TN to cover the publishing fee)

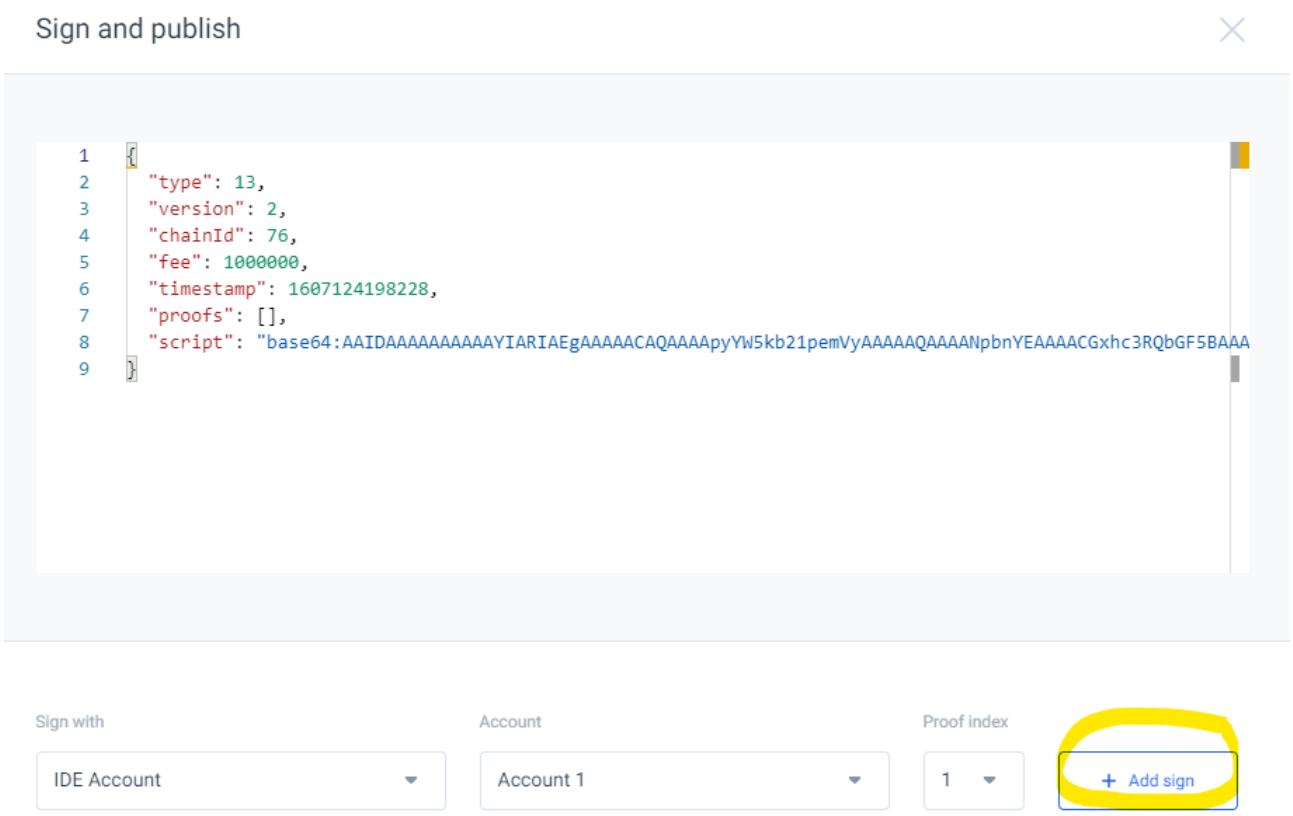




- Then click on ALT + N to open a new file script for dapps :

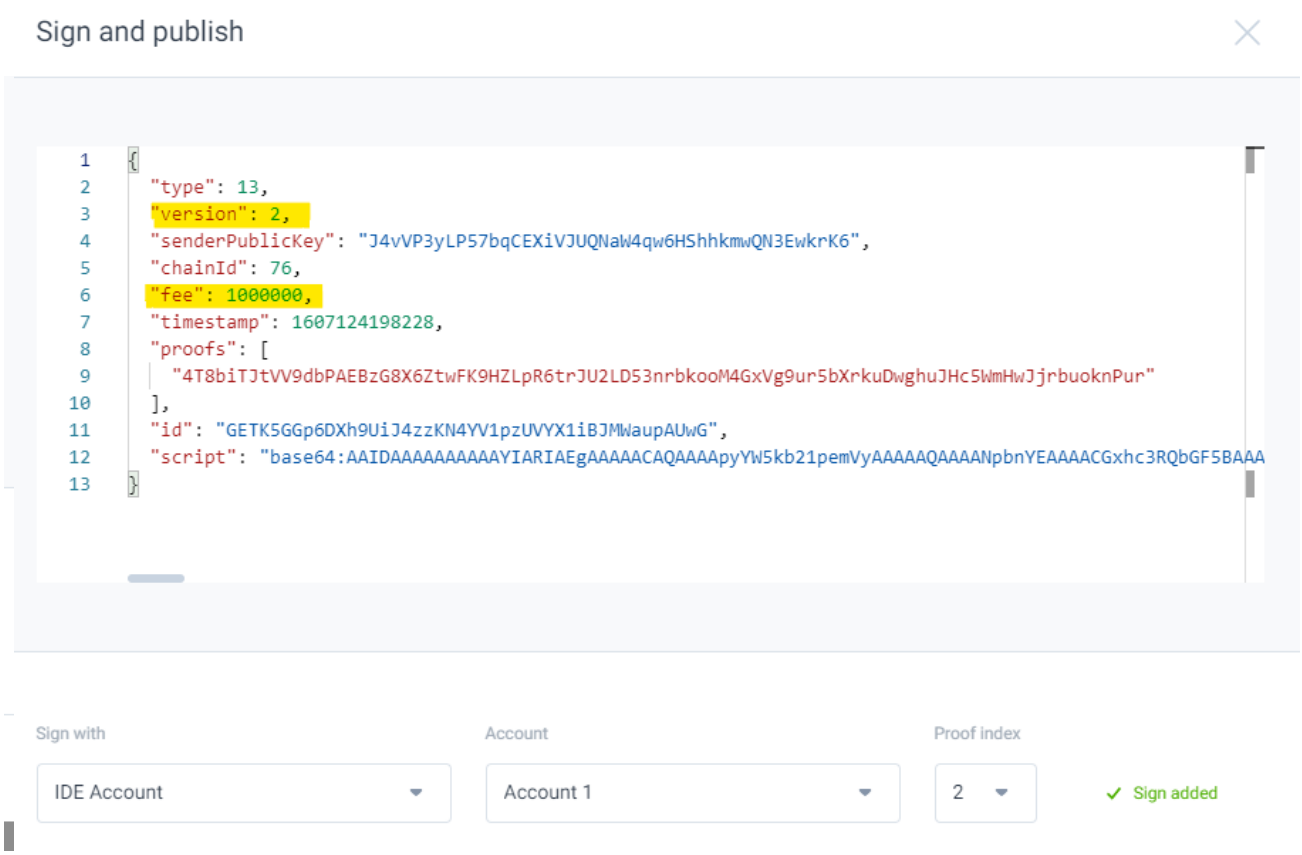
-And write your script or copy and paste a file from external source, the script will be automatically compiled as you enter it, if the compilation is OK, you can then deploy it by clicking on the deploy button as shown bellow :

- Then click on add sign :

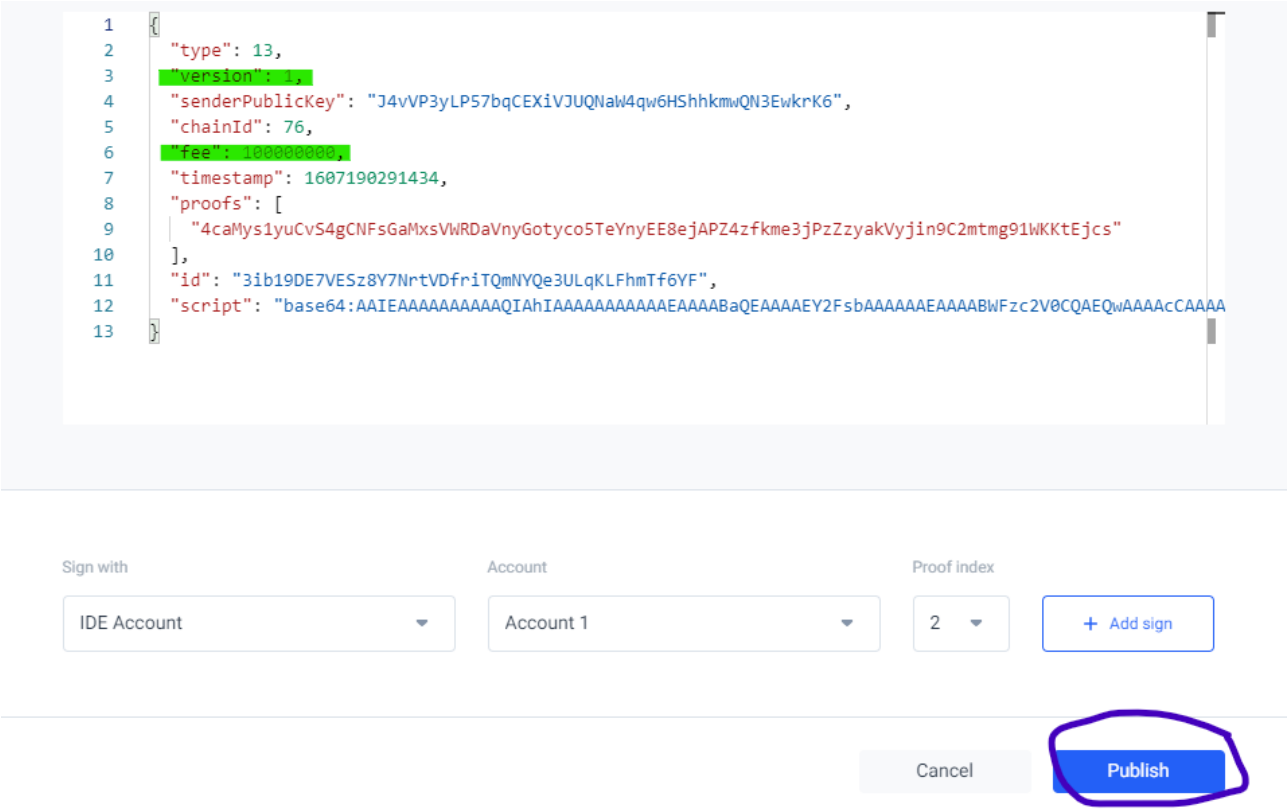


- Before publishing your dapp, change version 2 to 1 and fee 1000000 to 100000000

~~version 2~~ : version 1 and ~~fee 1000000~~ : fee 100000000



- At the end click on publish button to put your dapp on the blockchain :



Congratulations !
Your dapp is on the blockchain now,

A DAPP SCRIPT EXAMPLE

here is how a dapp script looks like :

```
{-# STDLIB_VERSION 4 #-}
{-# CONTENT_TYPE DAPP #-}
{-# SCRIPT_TYPE ACCOUNT #-}

@Callable(i)
func call() = {
 let asset = Issue("Asset", "", 1, 0, true, unit, 0)
 let assetId = asset.calculateAssetId()

 # Script execution results
 # More details in docs: https://docs.wavesplatform.com/en/ride/functions/callable-function#callable-functions-in-standard-library-v4
 [
  BinaryEntry("bin", base58''), # base16, base58, base64 or any other ByteVector values
  BooleanEntry("bool", true),
  IntegerEntry("int", 1),
  StringEntry("str", ""),
  DeleteEntry("str"),
  asset,
  Reissue(assetId, 1, false),
  Burn(assetId, 1),
  ScriptTransfer(i.caller, 1, assetId)
 ]
}

@Verifier(tx)
func verify() = sigVerify(tx.bodyBytes, tx.proofs[0], tx.senderPublicKey)
```

Do not use your own TN wallet to publish a Dapp, you might lose your assets, use a separate account for that !