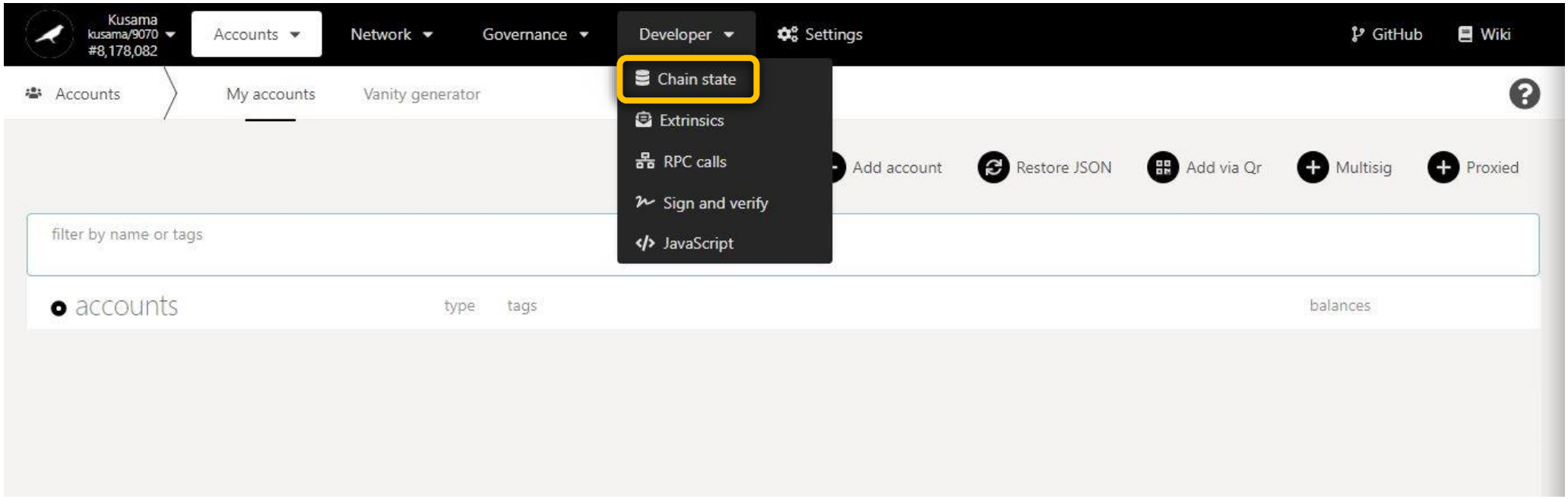
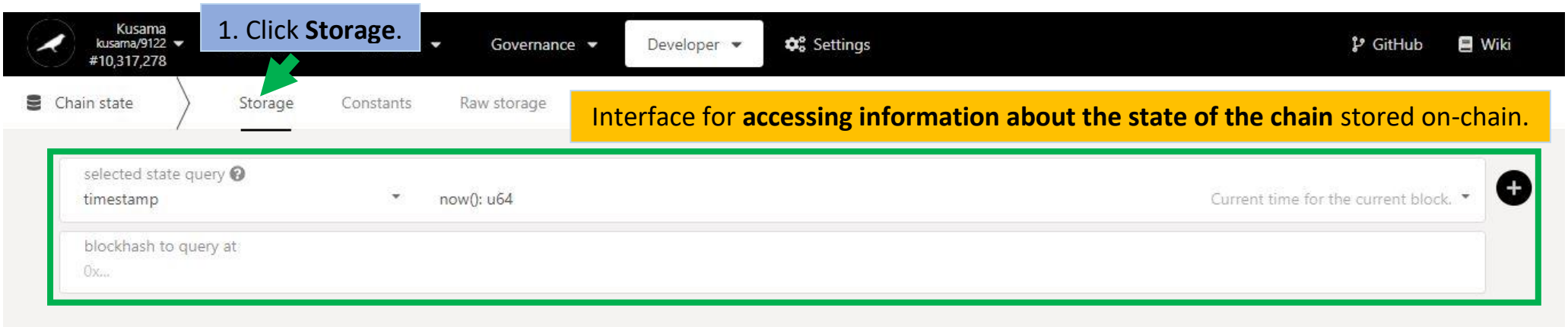
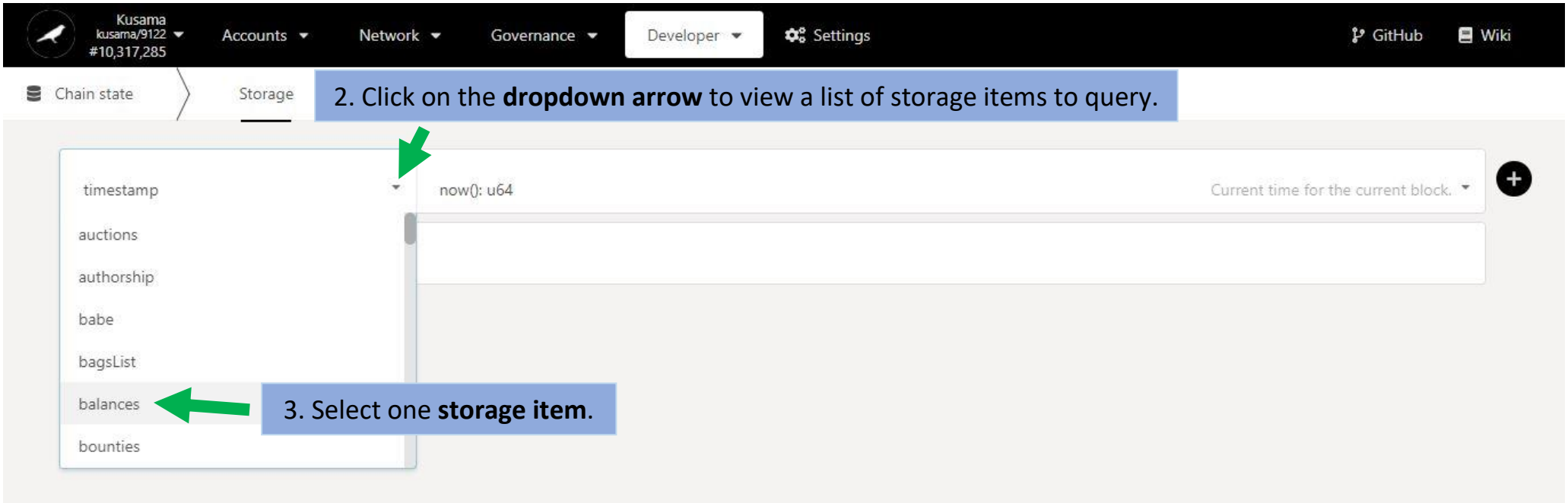
PART VIII: Developer

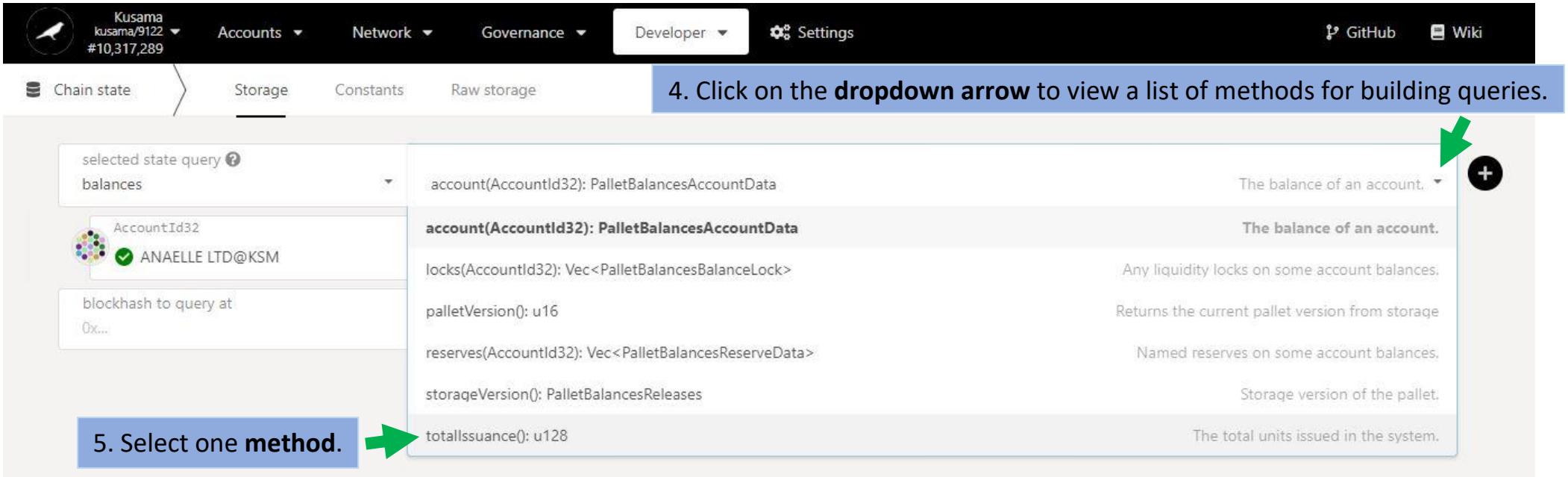
1. **Chain state:** Make API calls to query on-chain data.

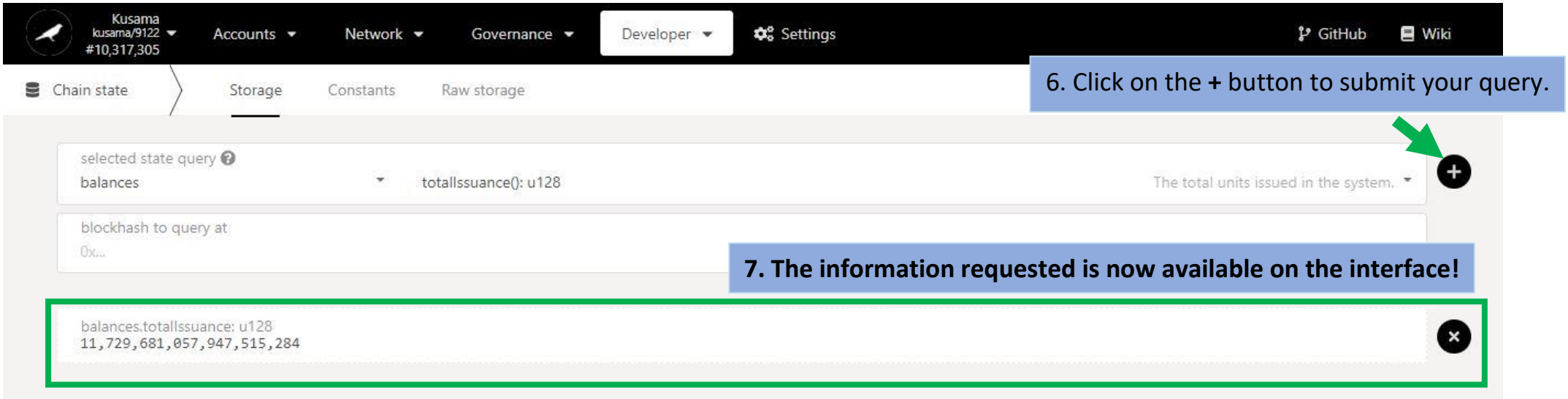


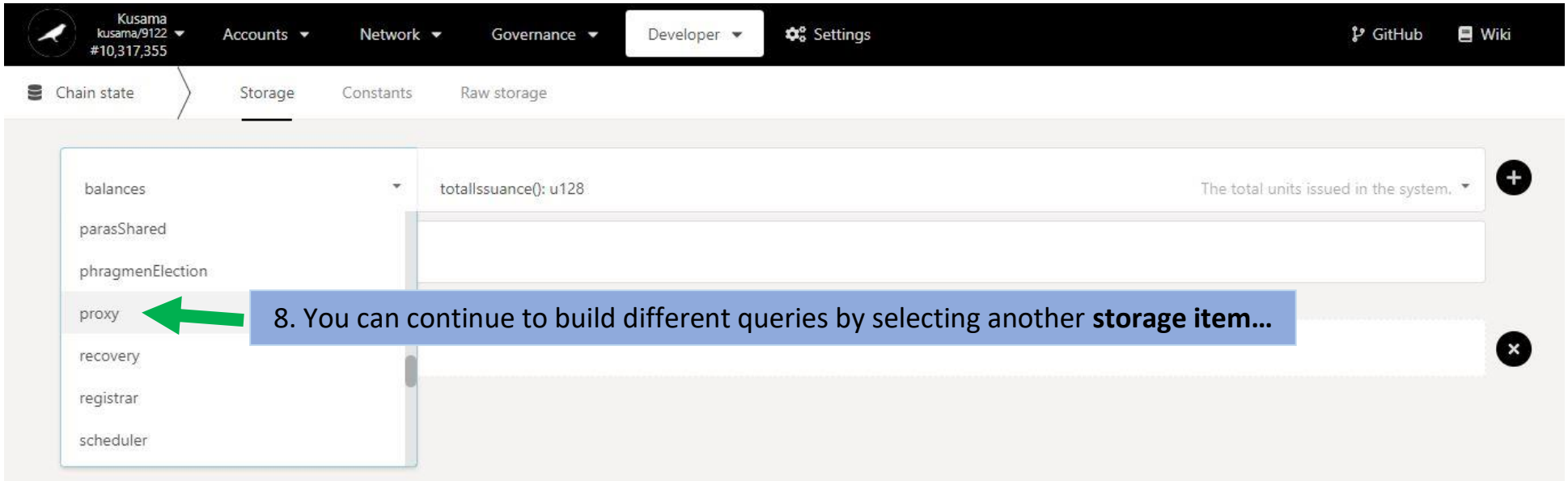
a) View storage items.

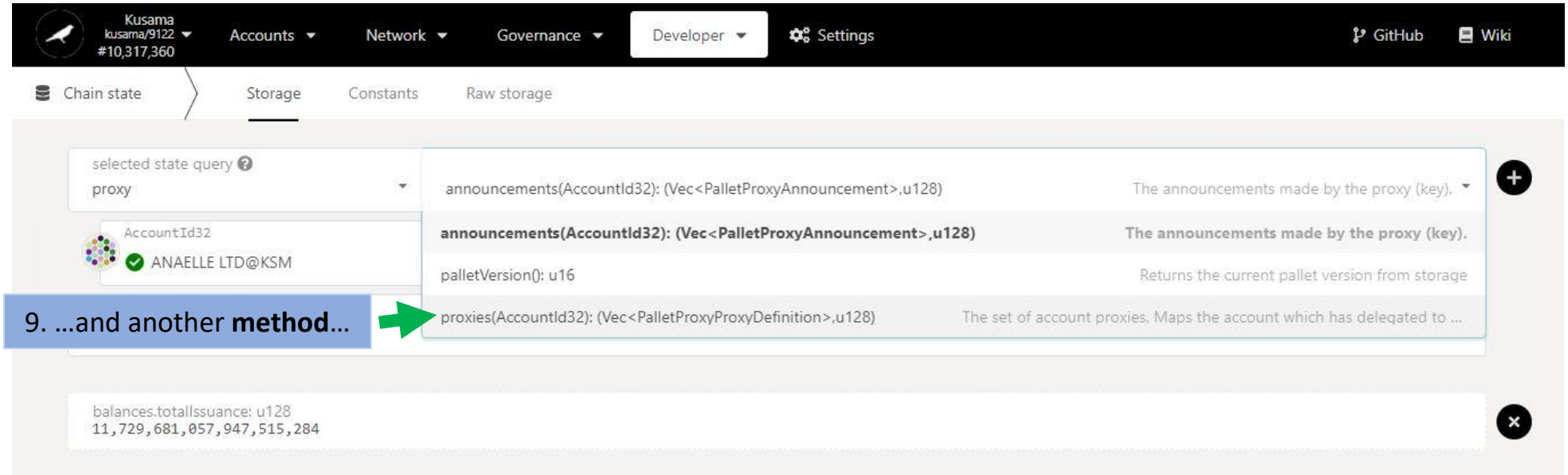


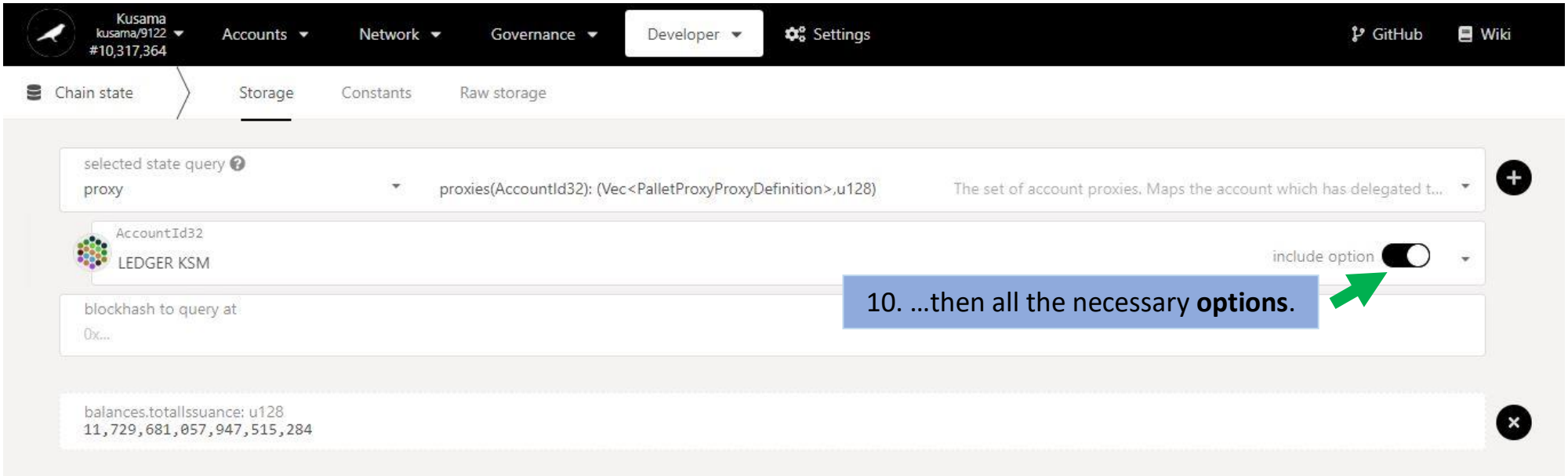



Kusama
kusama/9122
#10,317,289
Accounts
Network
Governance
Developer
Settings
GitHub
Wiki
Chain state
Storage
Constants
Raw storage
4. Click on the **dropdown arrow** to view a list of methods for building queries.
selected state query
balances
AccountId32
ANAELLE LTD@KSM
blockhash to query at
0x...
account(AccountId32): PalletBalancesAccountData
The balance of an account.
account(AccountId32): PalletBalancesAccountData
The balance of an account.
locks(AccountId32): Vec<PalletBalancesBalanceLock>
Any liquidity locks on some account balances.
palletVersion(): u16
Returns the current pallet version from storage
reserves(AccountId32): Vec<PalletBalancesReserveData>
Named reserves on some account balances.
storageVersion(): PalletBalancesReleases
Storage version of the pallet.
totalIssuance(): u128
The total units issued in the system.
5. Select one **method**.


Kusama
kusama/9122
#10,317,305
Accounts
Network
Governance
Developer
Settings
GitHub
Wiki
Chain state
Storage
Constants
Raw storage
6. Click on the **+** button to submit your query.
selected state query
balances
totalIssuance(): u128
The total units issued in the system.
blockhash to query at
0x...
7. The information requested is now available on the interface!
balances.totalIssuance: u128
11,729,681,057,947,515,284


Kusama
kusama/9122
#10,317,355
Accounts
Network
Governance
Developer
Settings
GitHub
Wiki
Chain state
Storage
Constants
Raw storage
balances
parasShared
phragmenElection
proxy
recovery
registrar
scheduler
totalIssuance(): u128
The total units issued in the system.
8. You can continue to build different queries by selecting another **storage item…**


Kusama
kusama/9122
#10,317,360
Accounts
Network
Governance
Developer
Settings
GitHub
Wiki
Chain state
Storage
Constants
Raw storage
selected state query
proxy
AccountId32
ANAELLE LTD@KSM
announcements(AccountId32): (Vec<PalletProxyAnnouncement>,u128)
The announcements made by the proxy (key).
announcements(AccountId32): (Vec<PalletProxyAnnouncement>,u128)
The announcements made by the proxy (key).
palletVersion(): u16
Returns the current pallet version from storage
proxies(AccountId32): (Vec<PalletProxyProxyDefinition>,u128)
The set of account proxies. Maps the account which has delegated to ...
9. …and another **method**…
balances.totalIssuance: u128
11,729,681,057,947,515,284


Kusama
kusama/9122
#10,317,364
Accounts
Network
Governance
Developer
Settings
GitHub
Wiki
Chain state
Storage
Constants
Raw storage
selected state query
proxy
proxies(AccountId32): (Vec<PalletProxyProxyDefinition>,u128)
The set of account proxies. Maps the account which has delegated t...
AccountId32
LEDGER KSM
include option
10. …then all the necessary **options**.
blockhash to query at
0x...
balances.totalIssuance: u128
11,729,681,057,947,515,284

Kusama
kusama/9122
#10,317,369

Accounts | Network | Governance | Developer | Settings | GitHub | Wiki

Chain state | Storage | Constants | Raw storage

11. Click on the + button to submit your second query.

selected state query
proxy
proxies(AccountId32): (Vec<PalletProxyProxyDefinition>,u128)
The set of account proxies. Maps the account which has delegated t...

AccountId32
LEDGER KSM
include option

blockhash to query at
0x...

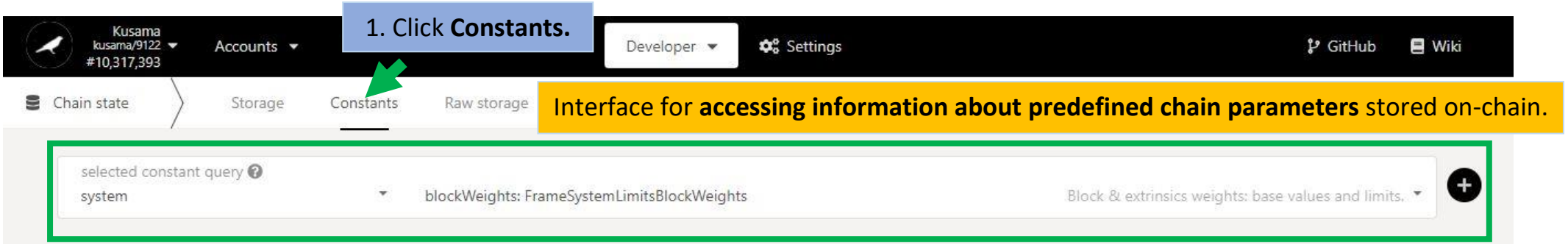
12. The new information requested is now stacked on the interface!

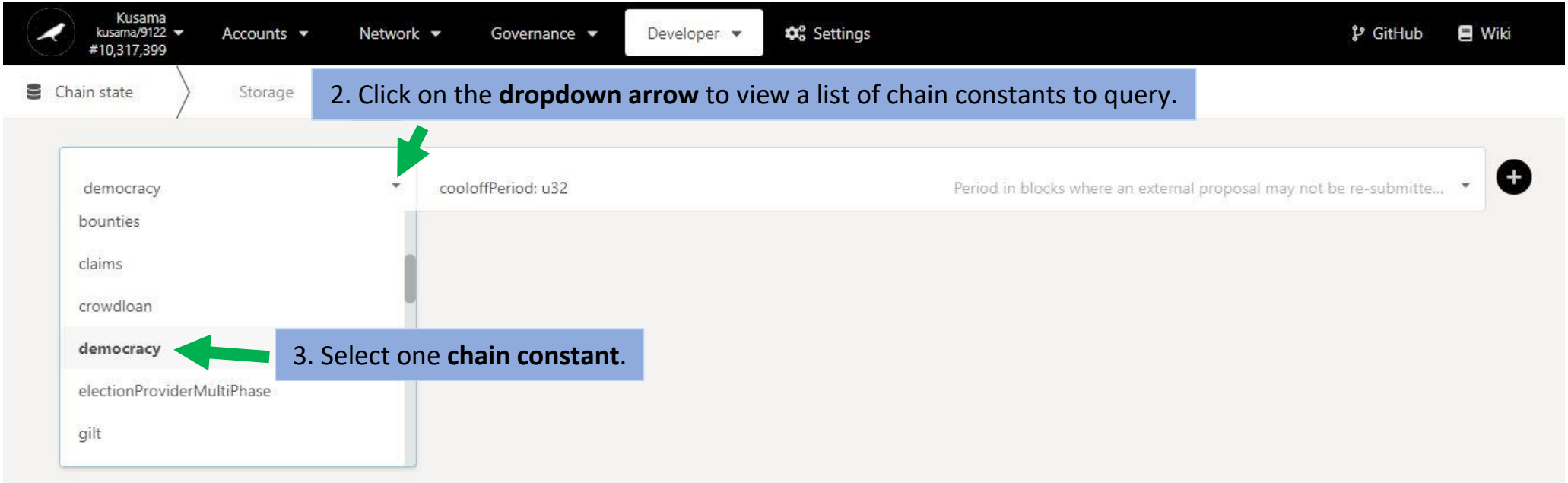
proxy.proxies: (Vec<PalletProxyProxyDefinition>,u128)

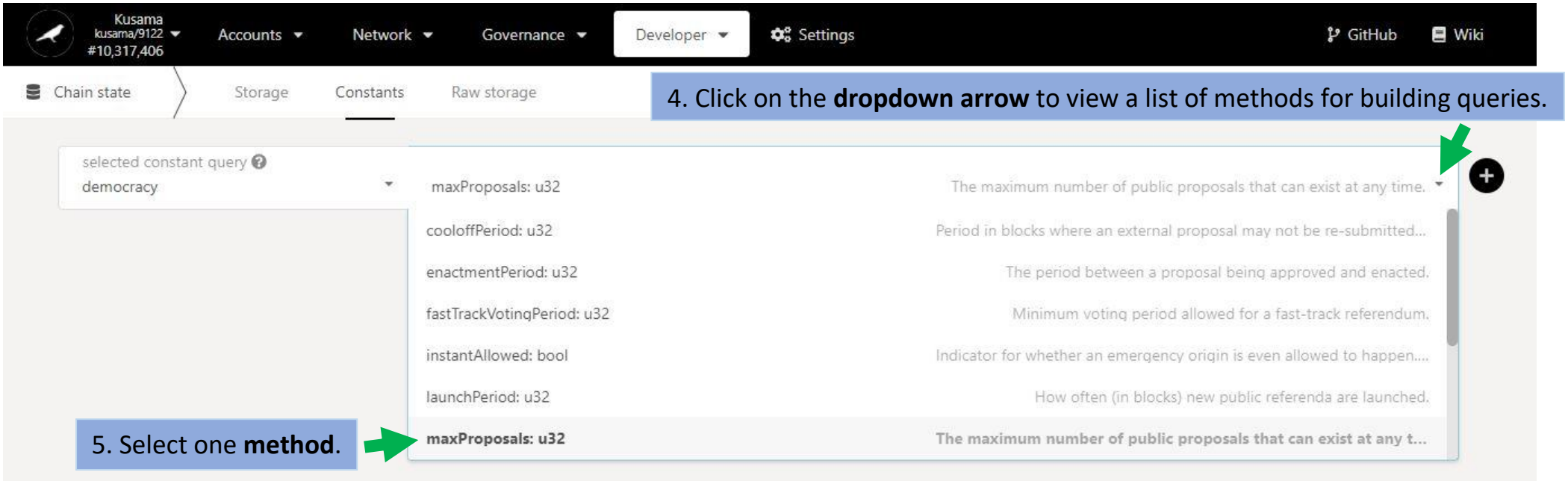
```
[
  [
    {
      delegate: HjcErRijmpoiBiKEHT3edPXM3NFycJogVwDPuByNe7hv9Ae
      proxyType: NonTransfer
      delay: 0
    }
  ]
  66,803,331,300
]
```

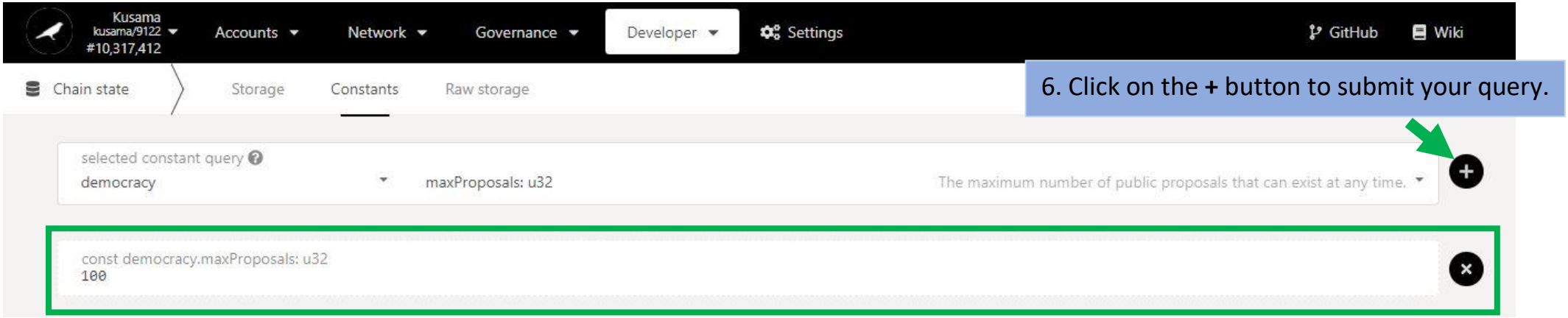
balances.totalIssuance: u128
11,729,681,057,937,515,284

b) View chain constants.

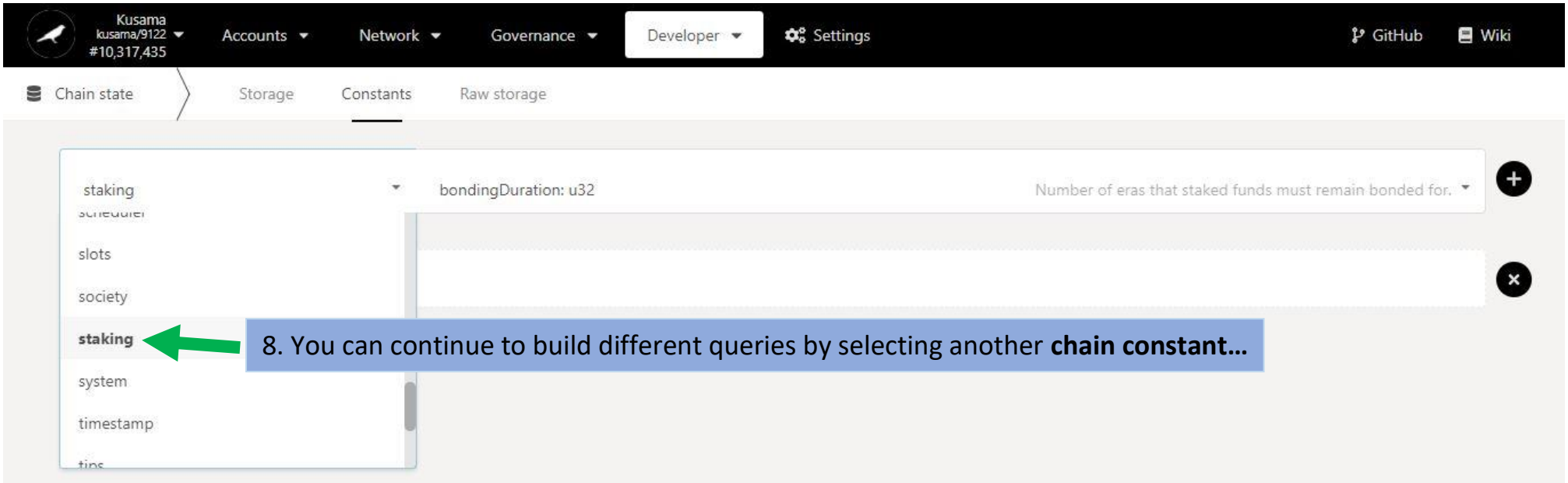


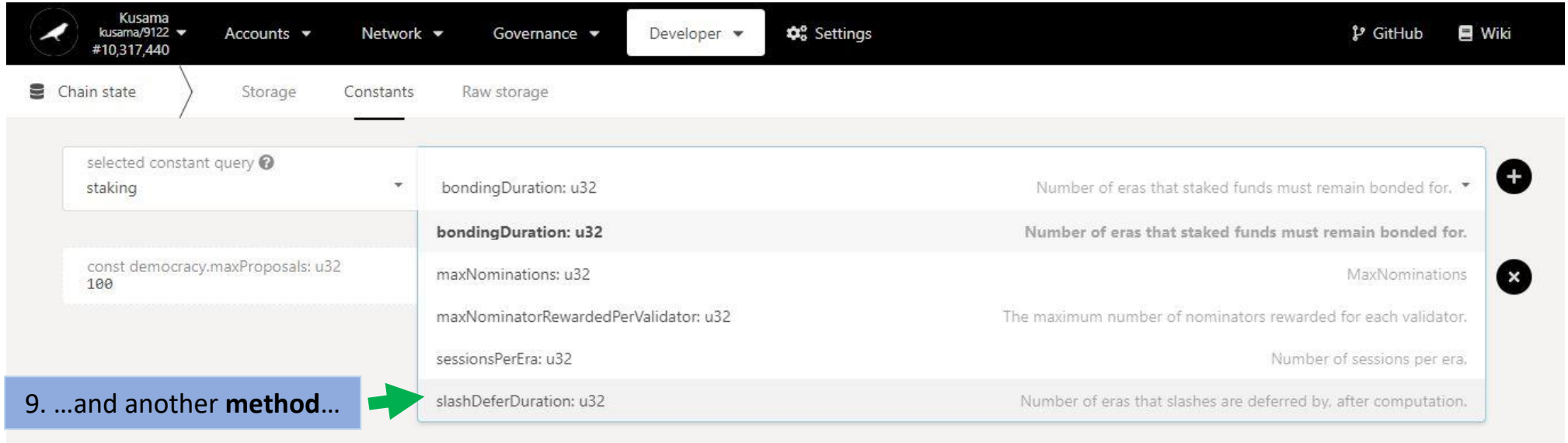


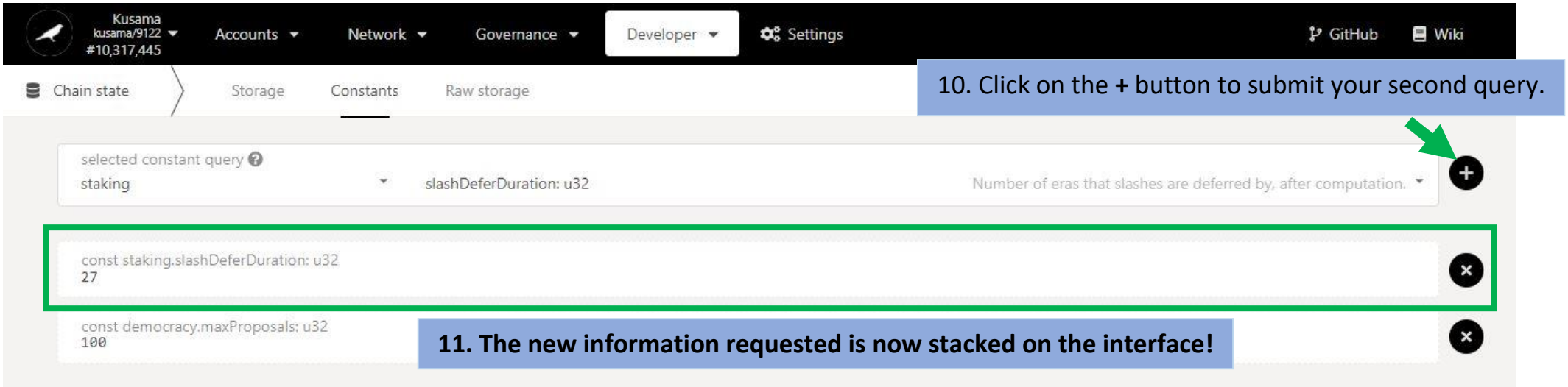




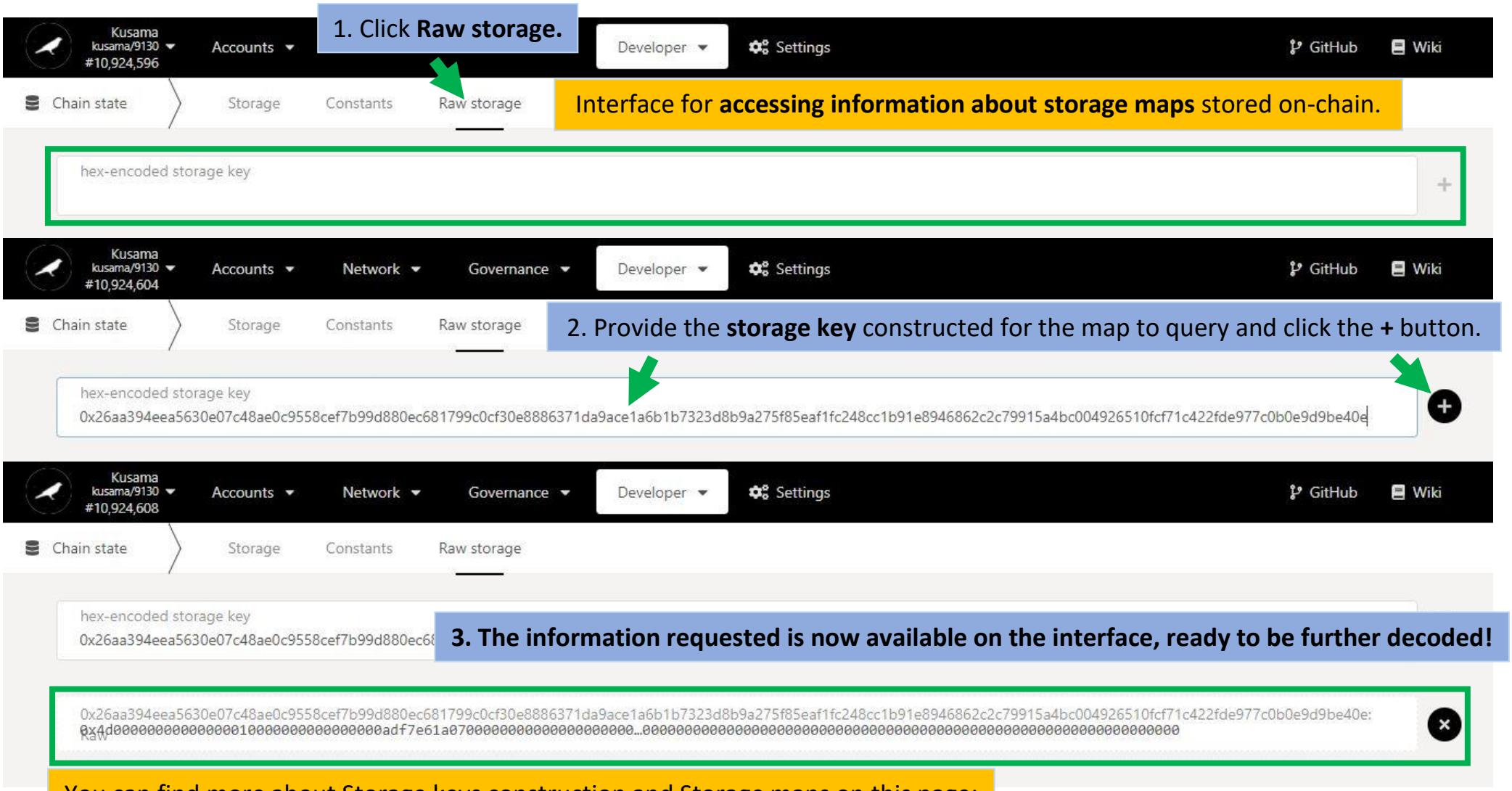
7. The information requested is now available on the interface!


Kusama
kusama/9122
#10,317,435
Accounts
Network
Governance
Developer
Settings
GitHub
Wiki
Chain state
Storage
Constants
Raw storage
staking
bondingDuration: u32
Number of eras that staked funds must remain bonded for.
slots
society
staking
system
timestamp
8. You can continue to build different queries by selecting another **chain constant…**


Kusama
kusama/9122
#10,317,440
Accounts
Network
Governance
Developer
Settings
GitHub
Wiki
Chain state
Storage
Constants
Raw storage
selected constant query
staking
bondingDuration: u32
Number of eras that staked funds must remain bonded for.
bondingDuration: u32
Number of eras that staked funds must remain bonded for.
const democracy.maxProposals: u32
100
maxNominations: u32
MaxNominations
maxNominatorRewardedPerValidator: u32
The maximum number of nominators rewarded for each validator.
sessionsPerEra: u32
Number of sessions per era.
slashDeferDuration: u32
Number of eras that slashes are deferred by, after computation.
9. …and another **method…**

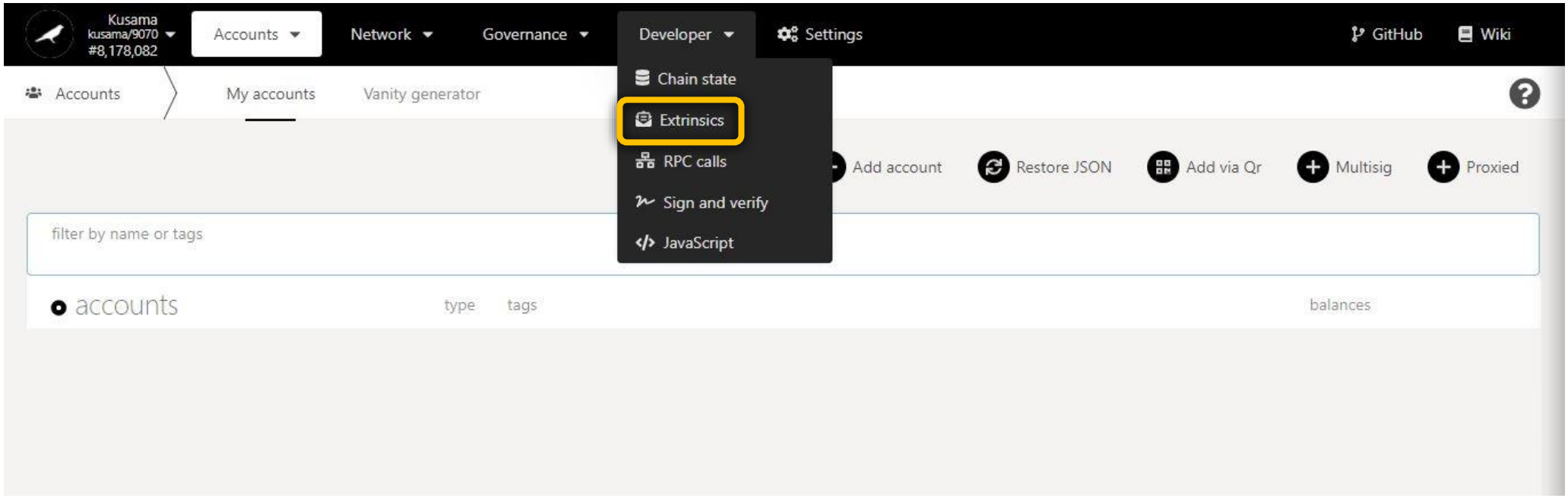

Kusama
kusama/9122
#10,317,445
Accounts
Network
Governance
Developer
Settings
GitHub
Wiki
Chain state
Storage
Constants
Raw storage
10. Click on the + button to submit your second query.
selected constant query
staking
slashDeferDuration: u32
Number of eras that slashes are deferred by, after computation.
const staking.slashDeferDuration: u32
27
const democracy.maxProposals: u32
100
11. The new information requested is now stacked on the interface!

c) Retrieve raw storage information.

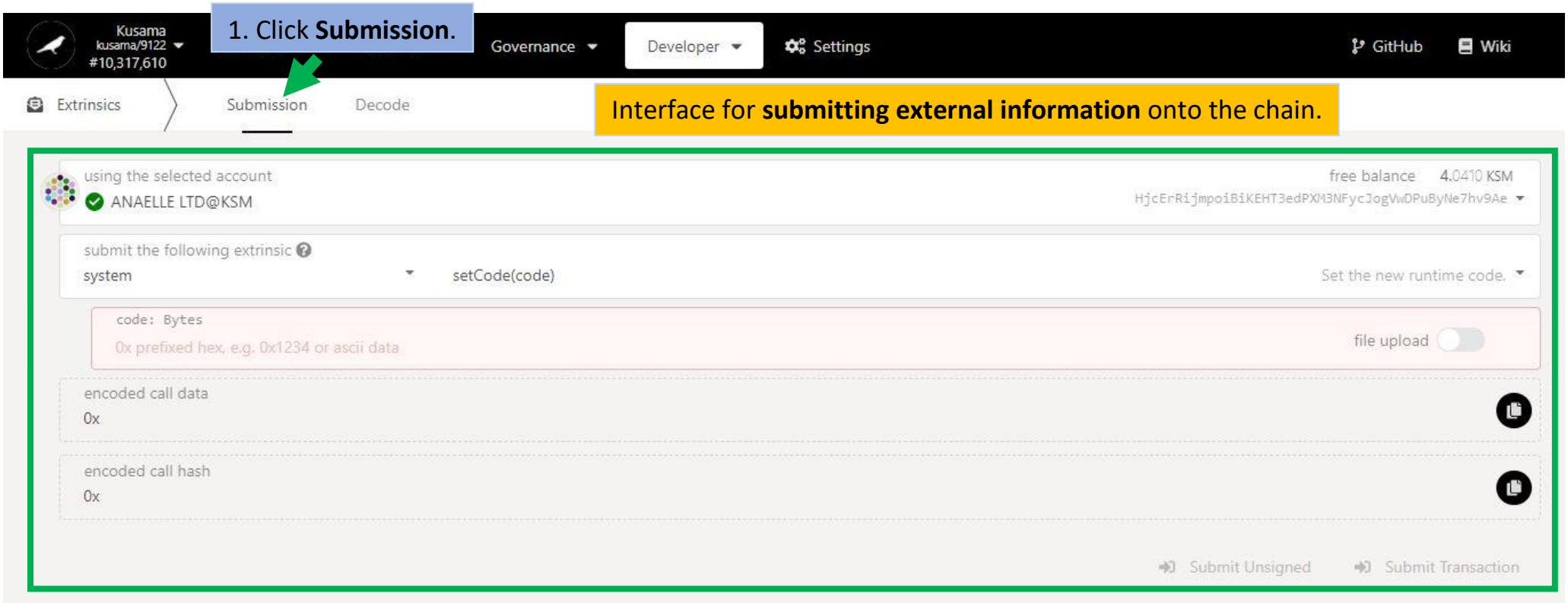


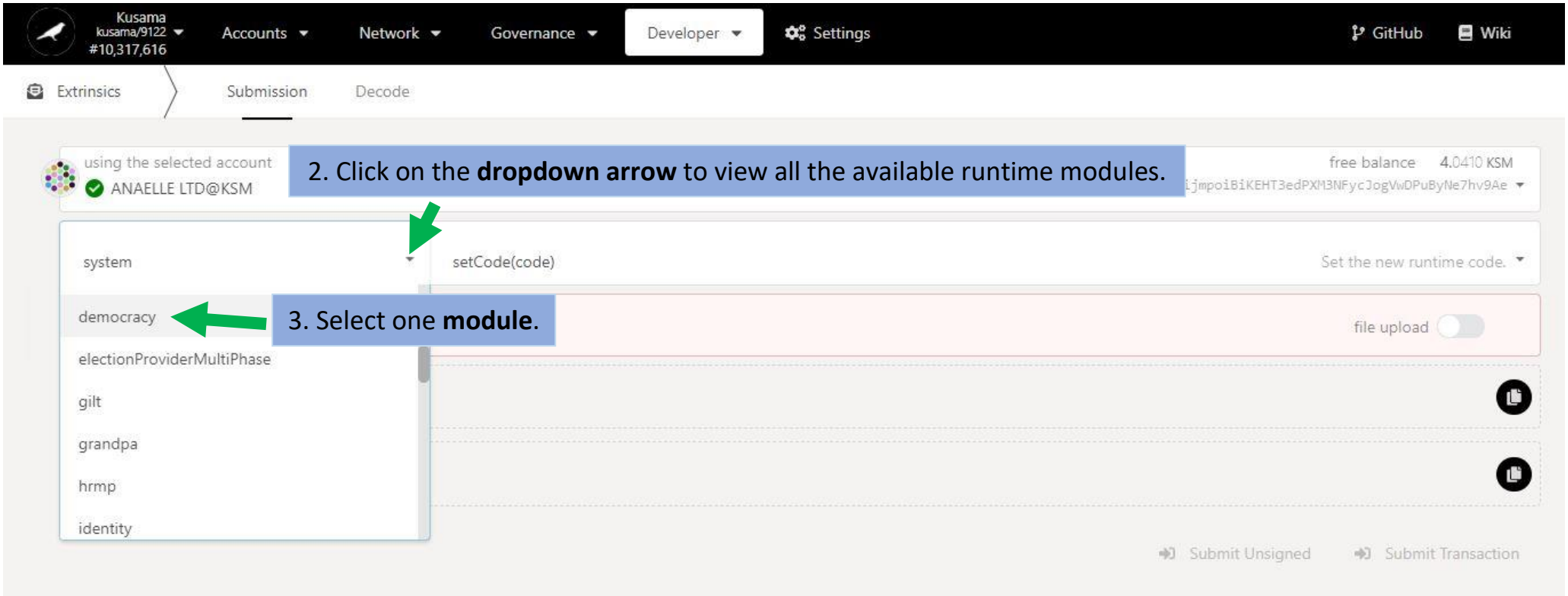
You can find more about Storage keys construction and Storage maps on this page:
https://www.shawntabrizi.com/substrate/transparent-keys-in-substrate/

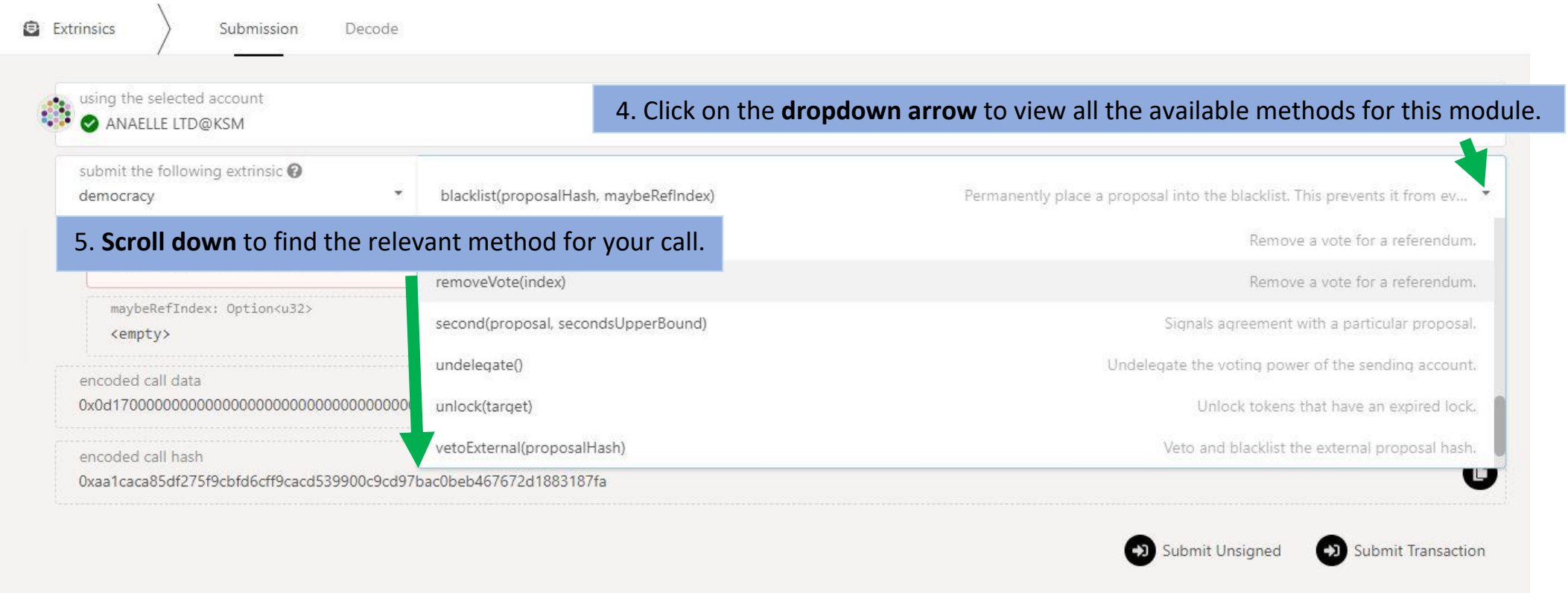
2. **Extrinsics:** Make API calls to submit data onto the chain directly.

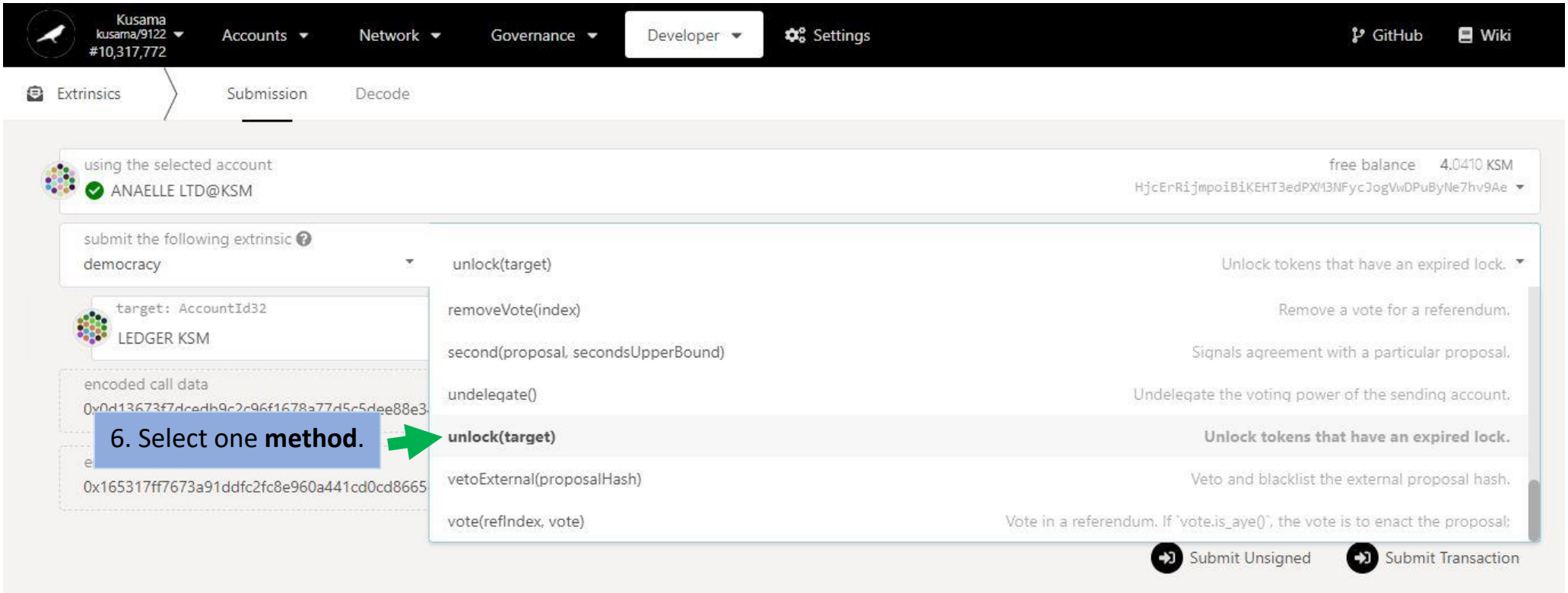


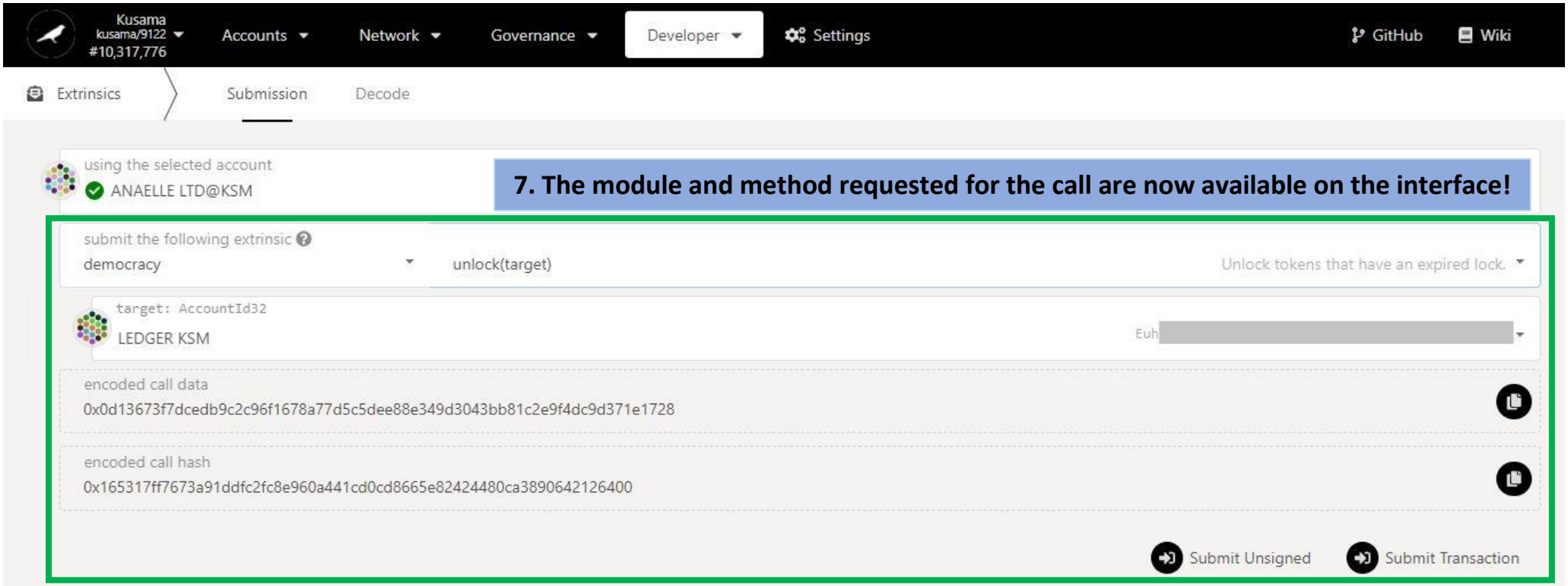
a) View runtime modules and methods.



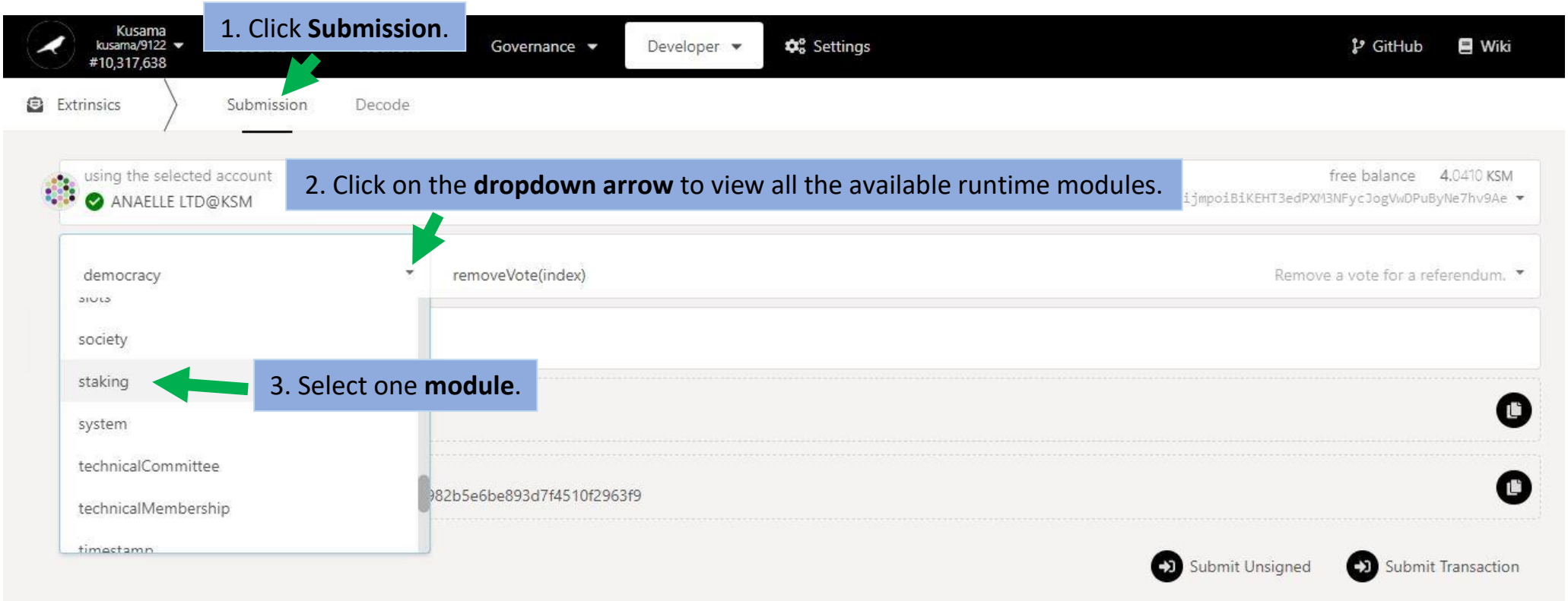

Kusama
kusama/9122
#10,317,616
Accounts
Network
Governance
Developer
Settings
GitHub
Wiki
Extrinsics
Submission
Decode
using the selected account
ANAELLE LTD@KSM
2. Click on the **dropdown arrow** to view all the available runtime modules.
free balance 4.0410 KSM
system
setCode(code)
Set the new runtime code.
democracy
3. Select one **module**.
file upload
electionProviderMultiPhase
gilt
grandpa
hrmp
identity
Submit Unsigned
Submit Transaction


Extrinsics
Submission
Decode
using the selected account
ANAELLE LTD@KSM
4. Click on the **dropdown arrow** to view all the available methods for this module.
submit the following extrinsic
democracy
blacklist(proposalHash, maybeRefIndex)
Permanently place a proposal into the blacklist. This prevents it from ev...
5. **Scroll down** to find the relevant method for your call.
Remove a vote for a referendum.
removeVote(index)
Remove a vote for a referendum.
maybeRefIndex: Option<u32>
<empty>
second(proposal, secondsUpperBound)
Signals agreement with a particular proposal.
undelegate()
Undelegate the voting power of the sending account.
encoded call data
unlock(target)
Unlock tokens that have an expired lock.
encoded call hash
vetoExternal(proposalHash)
Veto and blacklist the external proposal hash.
0xaa1caca85df275f9cbfd6cff9cacd539900c9cd97bac0beb467672d1883187fa
Submit Unsigned
Submit Transaction


Kusama
kusama/9122
#10,317,772
Accounts
Network
Governance
Developer
Settings
GitHub
Wiki
Extrinsics
Submission
Decode
using the selected account
ANAELLE LTD@KSM
free balance 4.0410 KSM
HjcErRijmpoiBiKEHT3edPXM3NFycJogVwDPuByNe7hv9Ae
submit the following extrinsic
democracy
unlock(target)
Unlock tokens that have an expired lock.
target: AccountId32
LEDGER KSM
removeVote(index)
Remove a vote for a referendum.
second(proposal, secondsUpperBound)
Signals agreement with a particular proposal.
undelegate()
Undelegate the voting power of the sending account.
encoded call data
6. Select one method.
unlock(target)
Unlock tokens that have an expired lock.
0x165317ff7673a91ddfc2fc8e960a441cd0cd8665
vetoExternal(proposalHash)
Veto and blacklist the external proposal hash.
vote(refIndex, vote)
Vote in a referendum. If `vote.is_aye()`, the vote is to enact the proposal;
Submit Unsigned
Submit Transaction


Kusama
kusama/9122
#10,317,776
Accounts
Network
Governance
Developer
Settings
GitHub
Wiki
Extrinsics
Submission
Decode
using the selected account
ANAELLE LTD@KSM
7. The module and method requested for the call are now available on the interface!
submit the following extrinsic
democracy
unlock(target)
Unlock tokens that have an expired lock.
target: AccountId32
LEDGER KSM
Euh
encoded call data
0x0d13673f7dcedb9c2c96f1678a77d5c5dee88e349d3043bb81c2e9f4dc9d371e1728
encoded call hash
0x165317ff7673a91ddfc2fc8e960a441cd0cd8665e82424480ca3890642126400
Submit Unsigned
Submit Transaction

b) Submit calls.



Kusama
kusama/9122
#10,317,651
Accounts
Network
Governance
Developer
Settings
GitHub
Wiki

Extrinsics
Submission
Decode

using the selected account
ANAELLE LTD@KSM

4. Click on the **dropdown arrow** to view all the available methods for this module.

submit the following extrinsic
staking

bond(controller, value, payee) — Take the origin account as a stash and lock up `value` of its balance. `c...

controller: MultiAddress (LookupSource)
Id

payoutStakers(validatorStash, era) — Pay out all the stakers behind a single validator for a single era.

reapStash(stash, numSlashingSpans) — Remove all data structure concerning a staker/stash once its balance is ...

5. Select one **method**.

rebond(value) — Rebond a portion of the stash scheduled to be unlocked.

value: Compact<u128> (BalanceOf)
0

scaleValidatorCount(factor) — Scale up the ideal number of validators by a factor.

setController(controller) — (Re-)set the controller of a stash.

payee: PalletStakingRewardDestination
Staked

setHistoryDepth(newHistoryDepth, eraItemsDeleted) — Set `HistoryDepth` value. This function will delete any history information

encoded call data
0x060000673f7dcedb9c2c96f1678a77d5c5dee88e349d3043bb81c2e9f4dc9d371e17280000

encoded call hash
0x78e8d1f63da23426056266c32fcb7788f0691c36bdfa59dd96d74e1d1239b4db

Kusama
kusama/9122
#10,317,658
Accounts
Network
Governance
Developer
Settings
GitHub
Wiki

Extrinsics
Submission
Decode

using the selected account
ANAELLE LTD@KSM
free balance 4.0410 KSM
HjcErRijmpoiBiKEHT3edPXM3NFycJogVwDPuByNe7hv9Ae

submit the following e
staking

6. Enter **the information required**.

Rebond a portion of the stash scheduled to be unlocked.

value: Compact<u128> (BalanceOf)
2

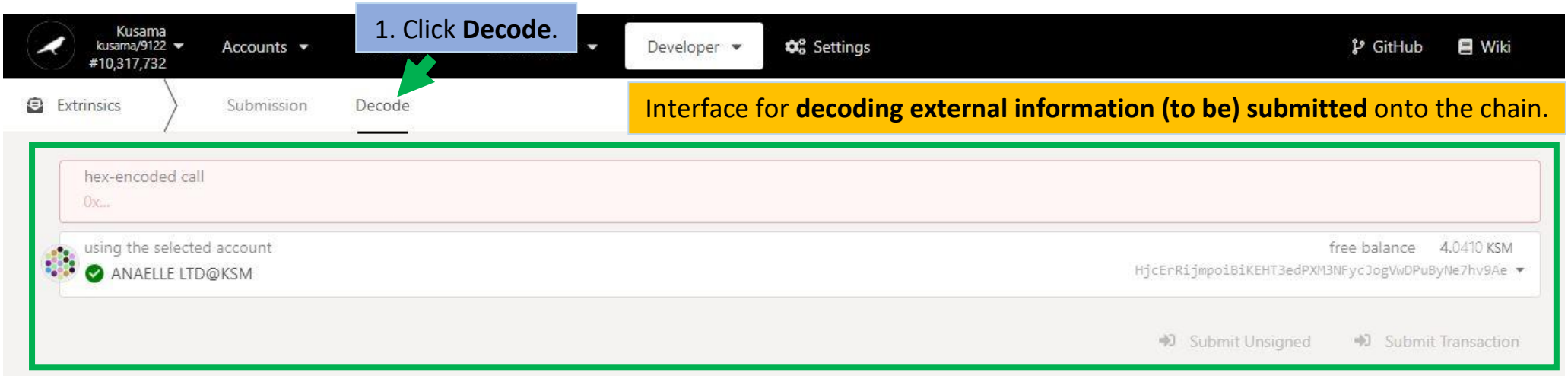
encoded call data
0x061308

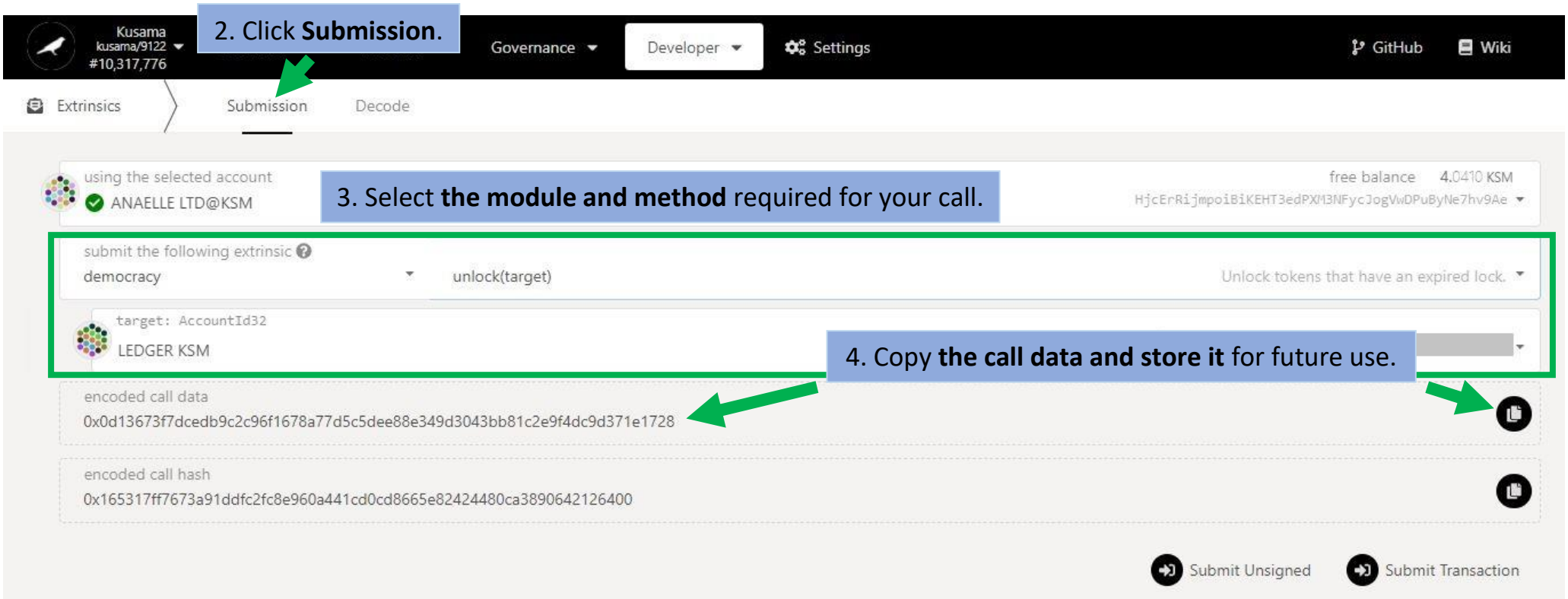
encoded call hash
0xd156c384d0451ac7ecc448c9252e8614be2f24a841a535203dcdd76d3063f1ed

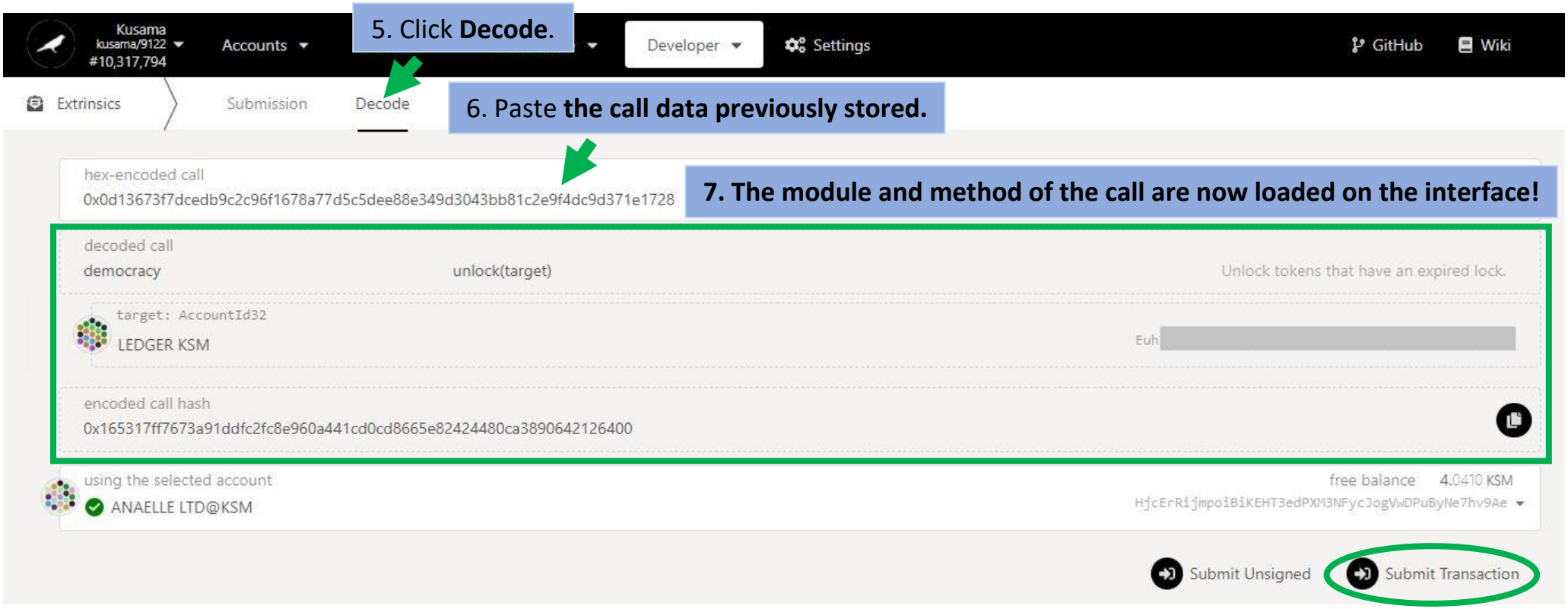
Submit Unsigned
Submit Transaction

7. Click on **Submit Transaction** to continue the procedure.

c) Decode hex-encoded calls.




Kusama
kusama/9122
#10,317,776
2. Click **Submission**.
Governance
Developer
Settings
GitHub
Wiki
Extrinsics
Submission
Decode
using the selected account
ANAELLE LTD@KSM
3. Select **the module and method** required for your call.
free balance 4.0410 KSM
HjcErRijmpoiBiKEHT3edPXM3NFycJogVwDPuByNe7hv9Ae
submit the following extrinsic
democracy
unlock(target)
Unlock tokens that have an expired lock.
target: AccountId32
LEDGER KSM
4. Copy **the call data and store it** for future use.
encoded call data
0x0d13673f7dcedb9c2c96f1678a77d5c5dee88e349d3043bb81c2e9f4dc9d371e1728
encoded call hash
0x165317ff7673a91ddfc2fc8e960a441cd0cd8665e82424480ca3890642126400
Submit Unsigned
Submit Transaction



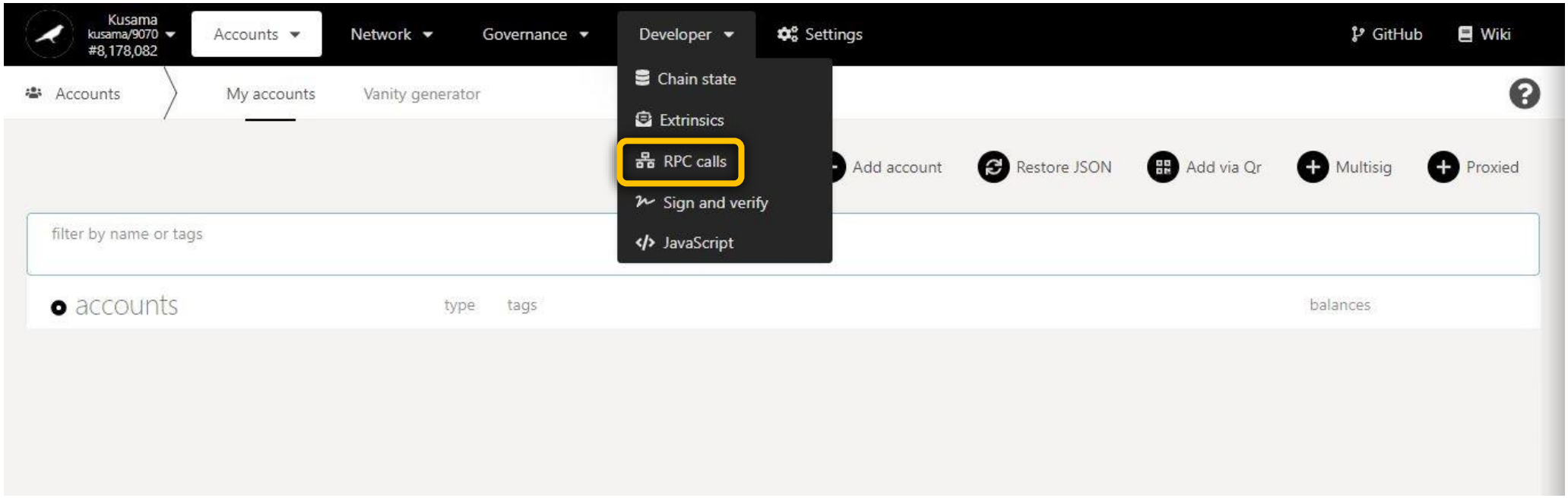
5. Click **Decode**.

6. Paste **the call data previously stored.**

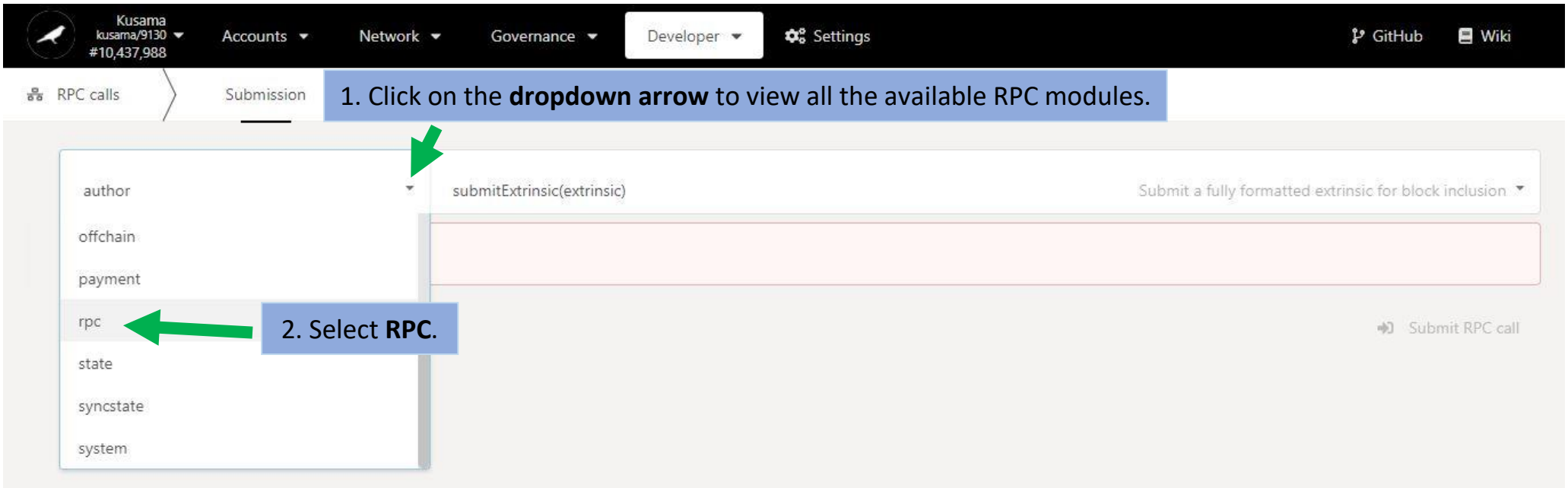
7. The module and method of the call are now loaded on the interface!

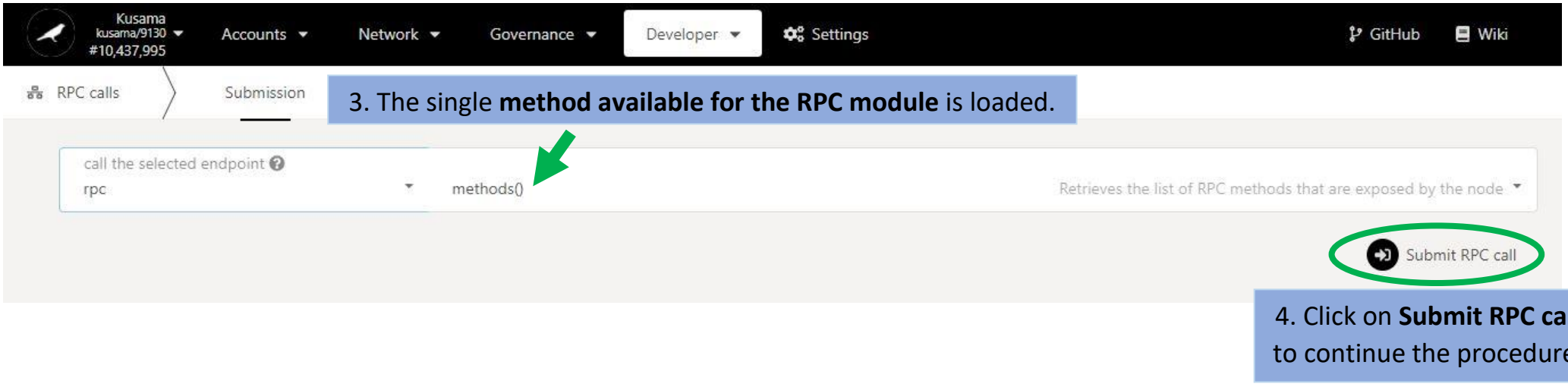
8. Click on **Submit Transaction** to continue the procedure.

3. **RPC calls:** Make RPC calls to submit data onto the chain remotely.



a) View JSON RPC modules and methods.





Kusama
kusama/9130
#10,437,999

Accounts | Network | Governance | Developer | Settings | GitHub | Wiki

RPC calls | Submission

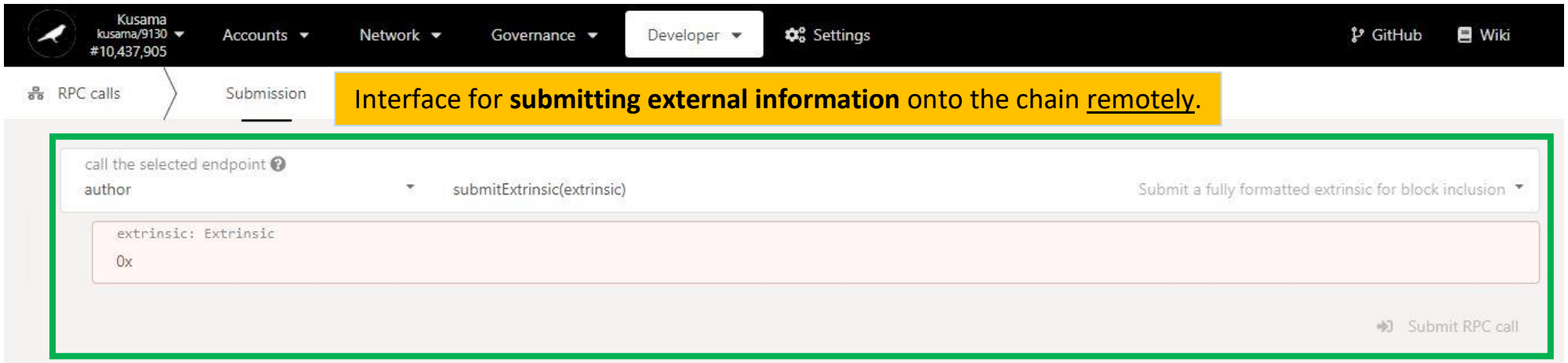
call the selected endpoint
rpc | methods() | Retrieves the list of RPC methods that are exposed by the node

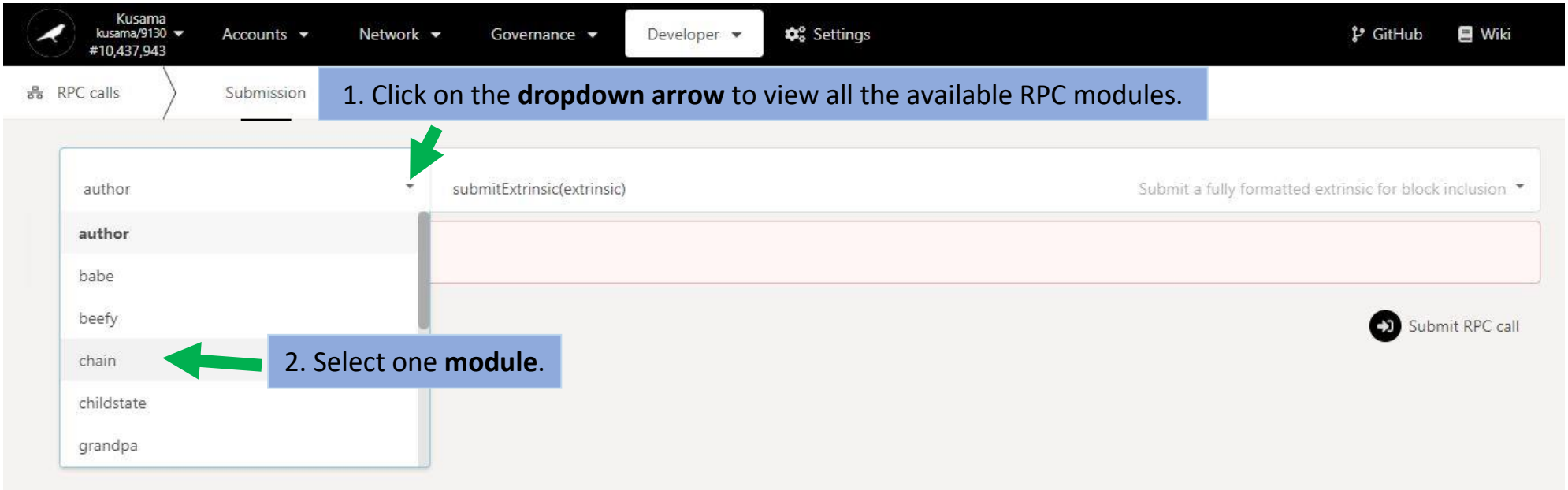
Submit RPC call

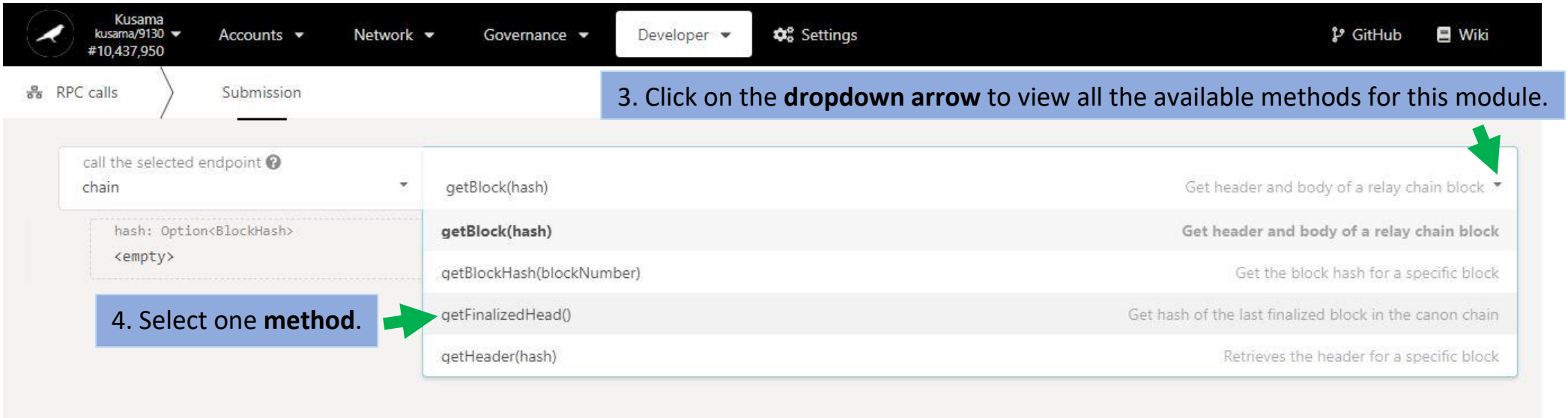
5. All RPC modules and methods available are now listed on the interface, ready to be used remotely!

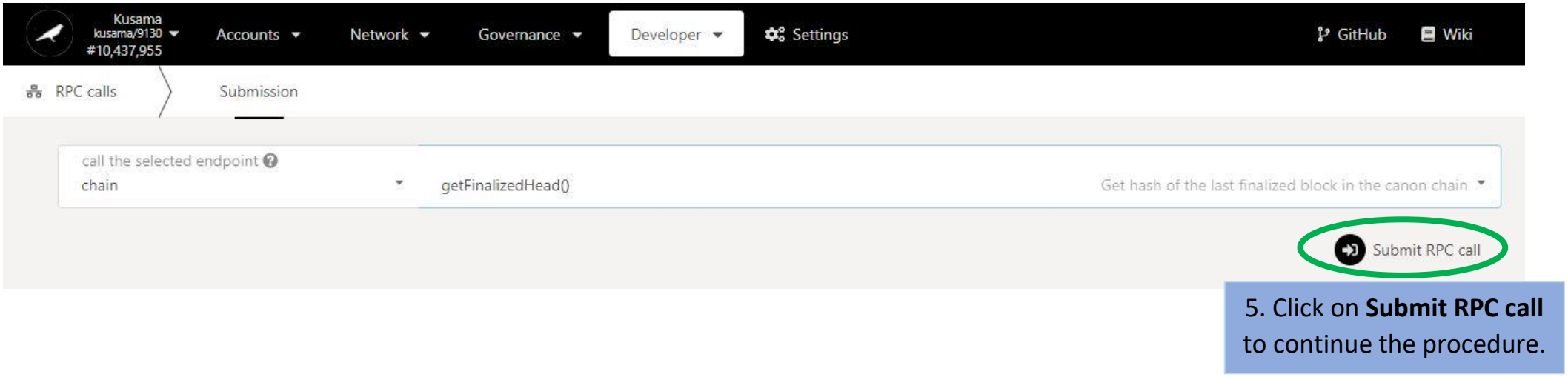
```
1: rpc.methods
{
  version: 1
  methods: [
    author_hasKey
    author_hasSessionKeys
    author_insertKey
    author_pendingExtrinsics
    author_removeExtrinsic
    author_rotateKeys
    author_submitAndWatchExtrinsic
    author_submitExtrinsic
    author_unsubscribesubmitAndWatchExtrinsic
    author_unwatchExtrinsic
    babe_epochAuthorship
    beefy_subscribeJustifications
    beefy_unsubscribeJustifications
    chain_getBlock
    chain_getBlockHash
    chain_getFinalisedHead
```

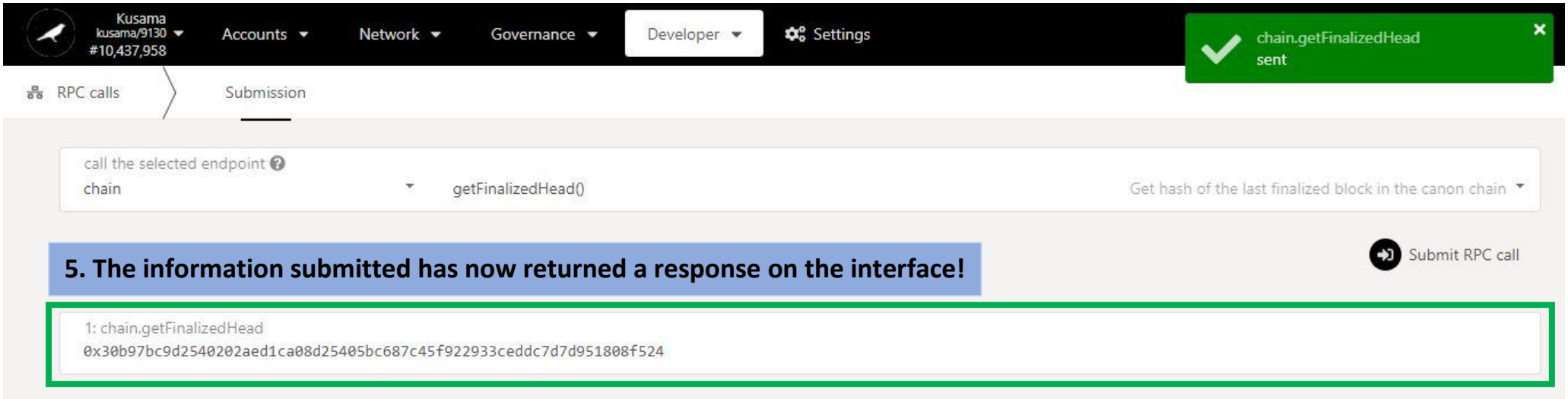
b) Submit RPC calls.





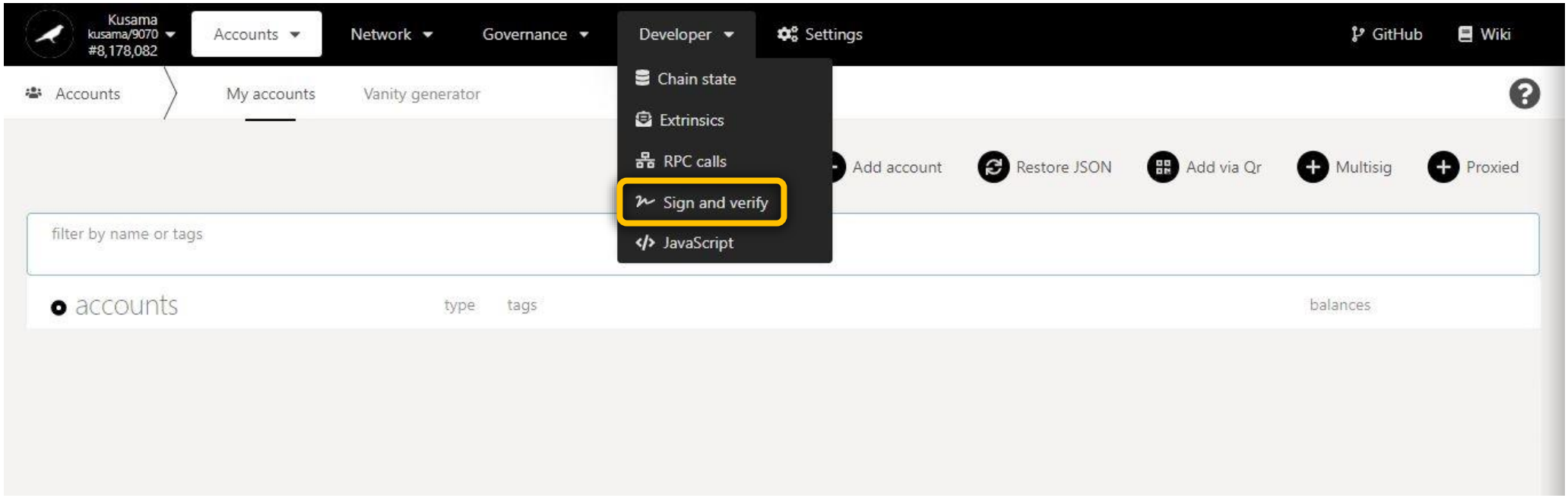

Kusama
kusama/9130
#10,437,950
Accounts
Network
Governance
Developer
Settings
GitHub
Wiki
RPC calls
Submission
3. Click on the **dropdown arrow** to view all the available methods for this module.
call the selected endpoint
chain
getBlock(hash)
Get header and body of a relay chain block
hash: Option<BlockHash>
<empty>
getBlock(hash)
Get header and body of a relay chain block
getBlockHash(blockNumber)
Get the block hash for a specific block
4. Select one **method**.
getFinalizedHead()
Get hash of the last finalized block in the canon chain
getHeader(hash)
Retrieves the header for a specific block


Kusama
kusama/9130
#10,437,955
Accounts
Network
Governance
Developer
Settings
GitHub
Wiki
RPC calls
Submission
call the selected endpoint
chain
getFinalizedHead()
Get hash of the last finalized block in the canon chain
Submit RPC call
5. Click on **Submit RPC call** to continue the procedure.

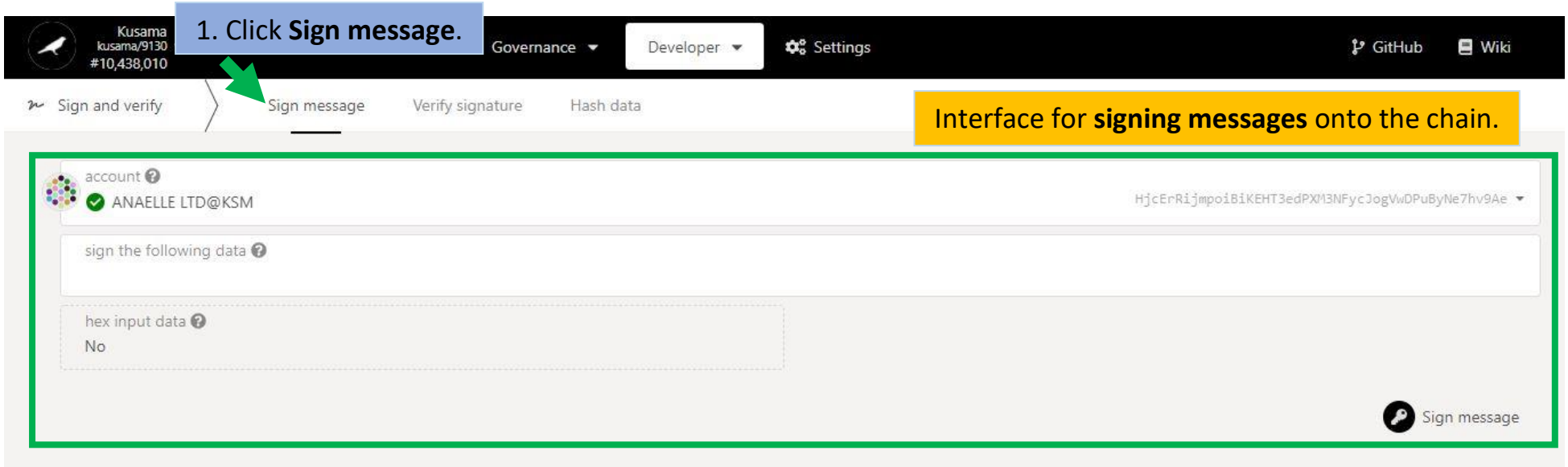


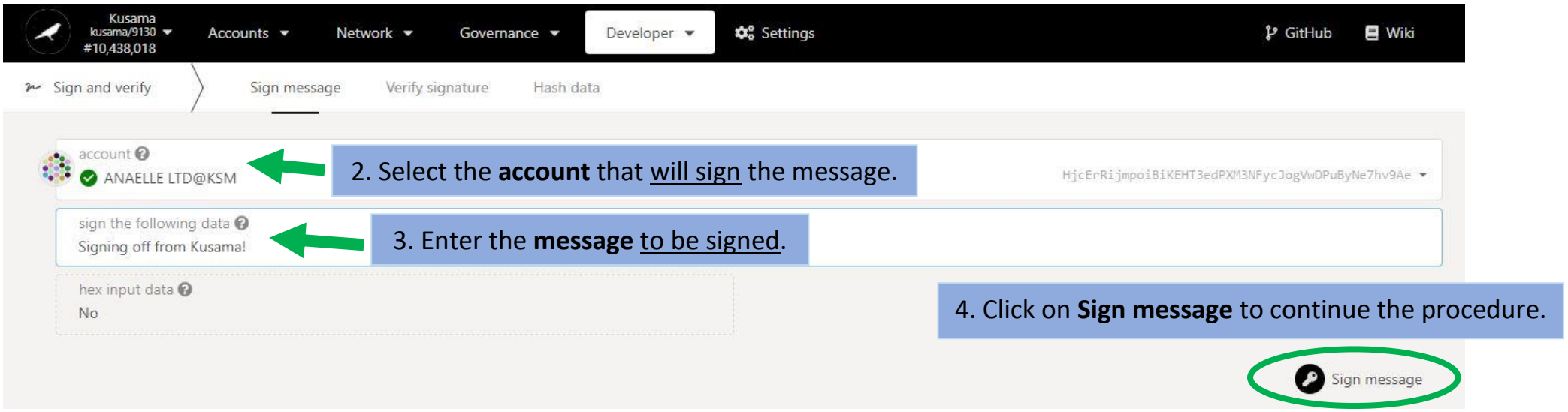
5. The information submitted has now returned a response on the interface!

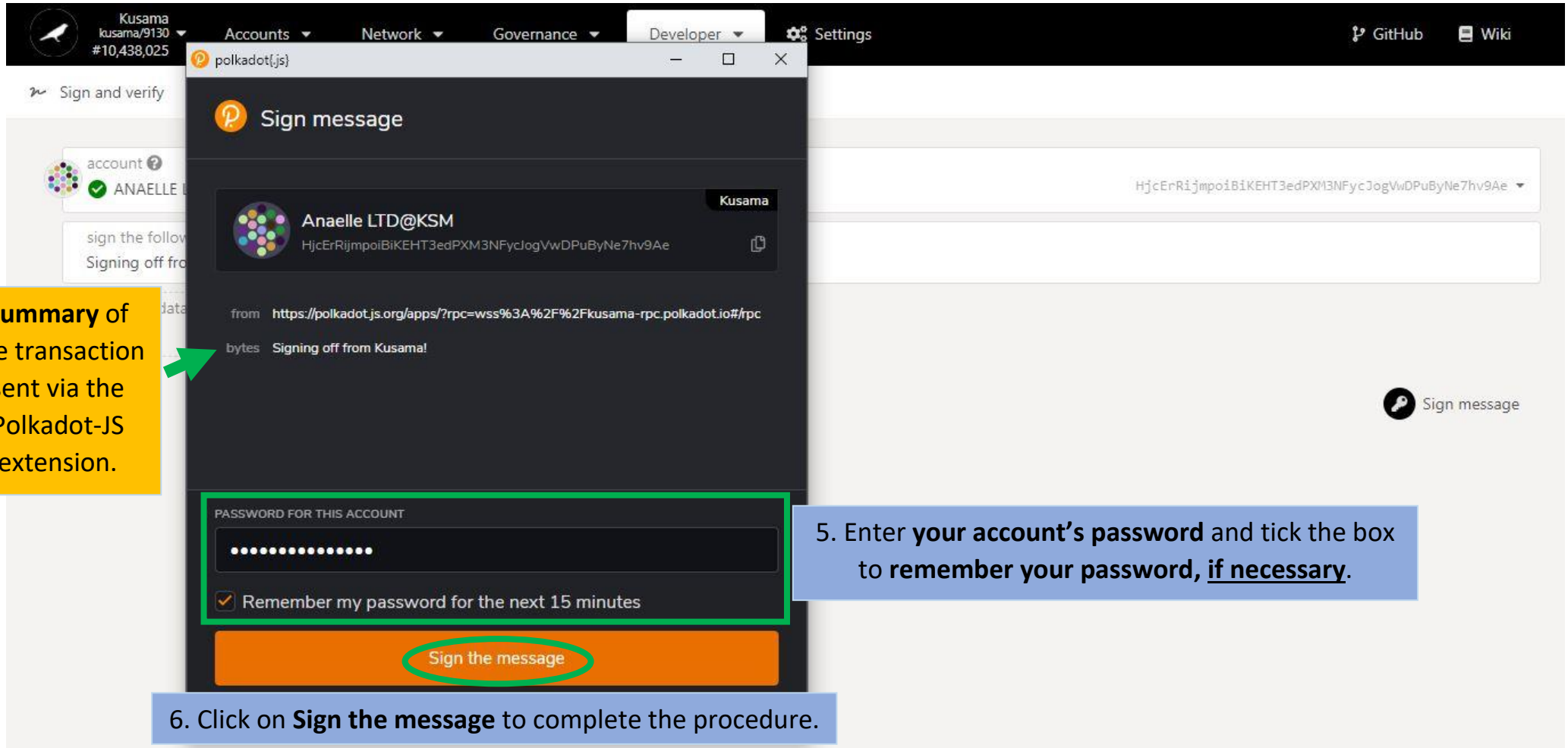
4. **Sign & Verify:** Sign messages onto the chain and verify on-chain signatures.



a) Sign messages.



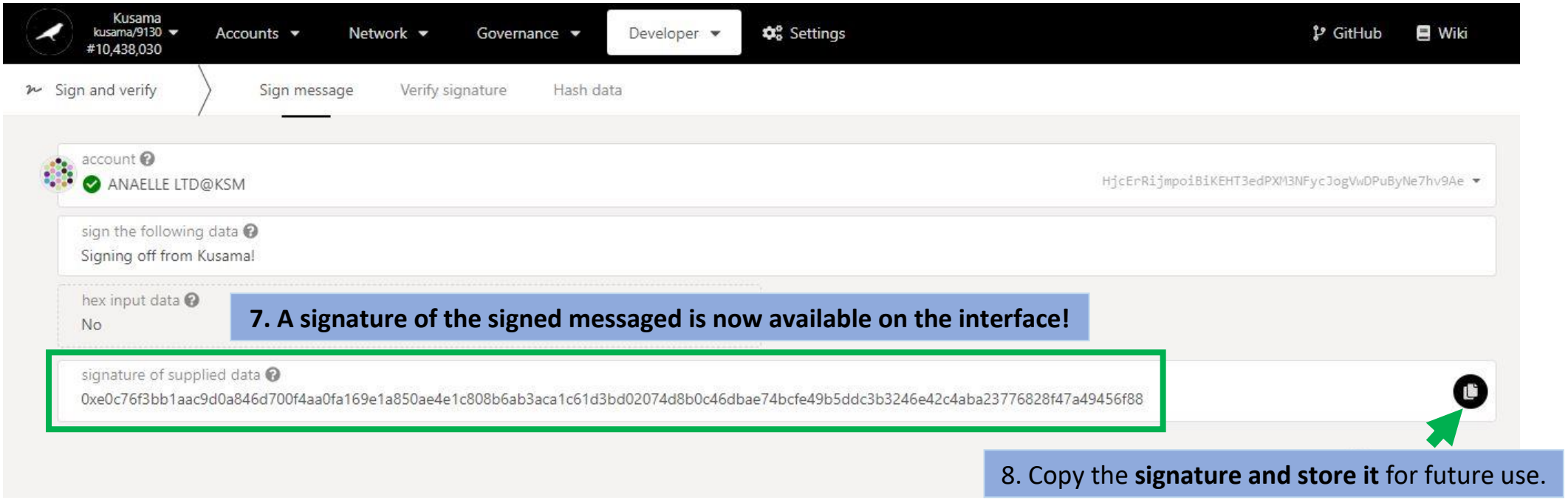




Summary of the transaction sent via the Polkadot-JS extension.

5. Enter **your account's password** and tick the box to **remember your password, <u>if necessary</u>**.

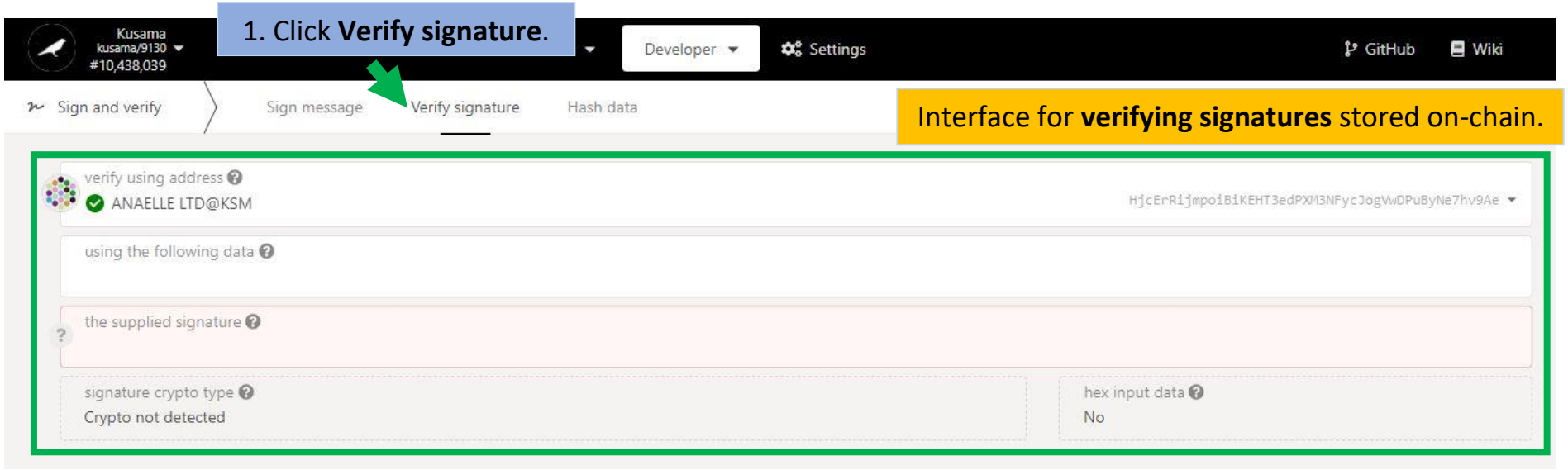
6. Click on **Sign the message** to complete the procedure.

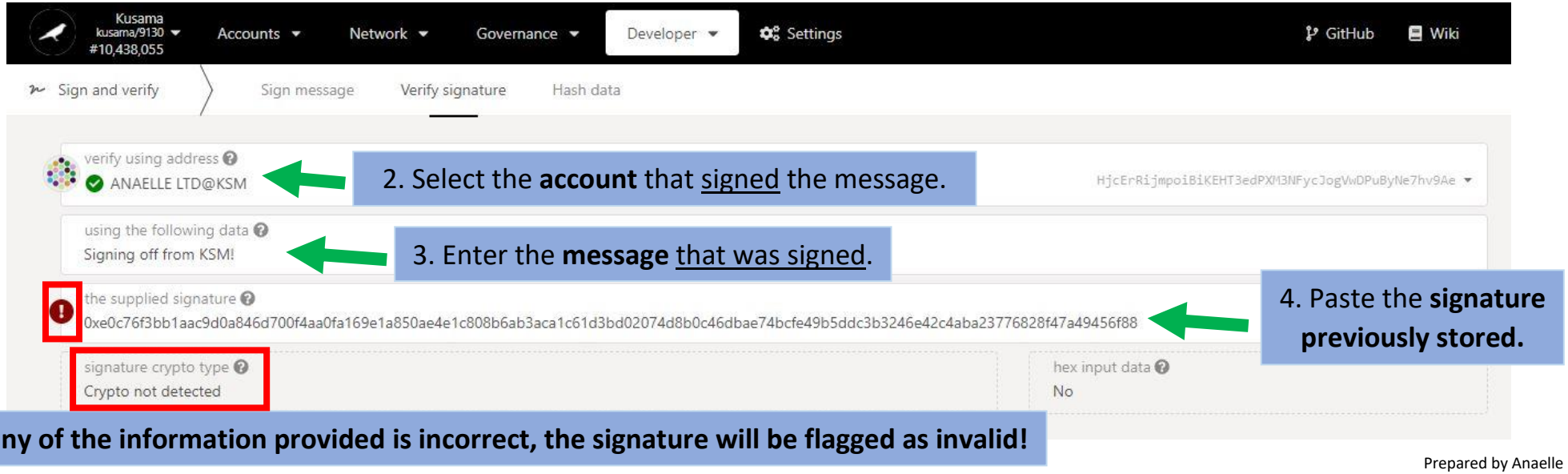


7. A signature of the signed messaged is now available on the interface!

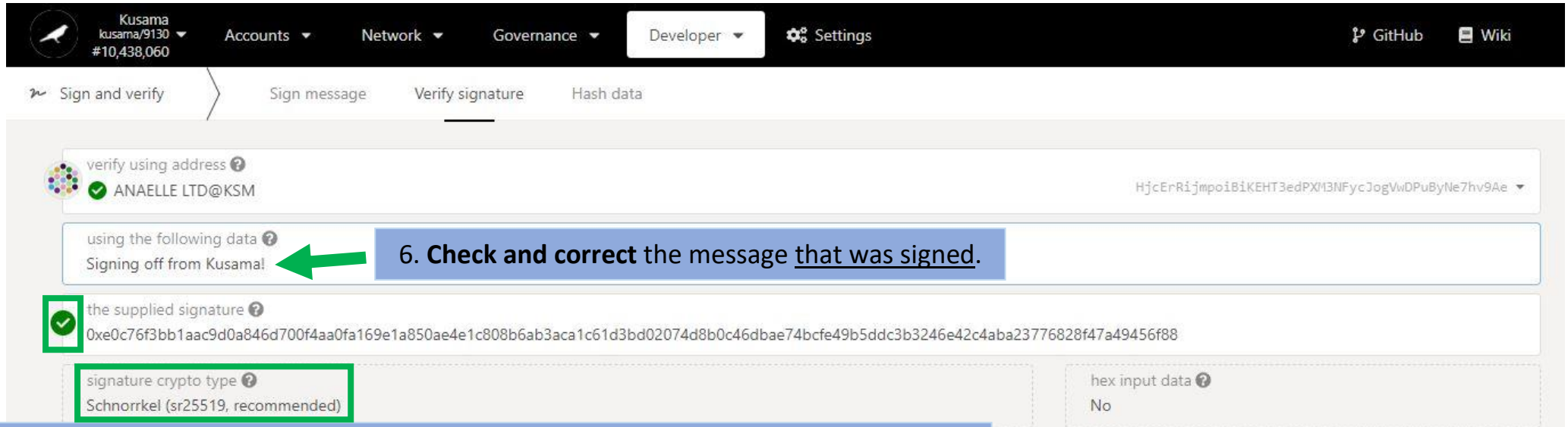
8. Copy the **signature and store it** for future use.

b) Verify signatures.





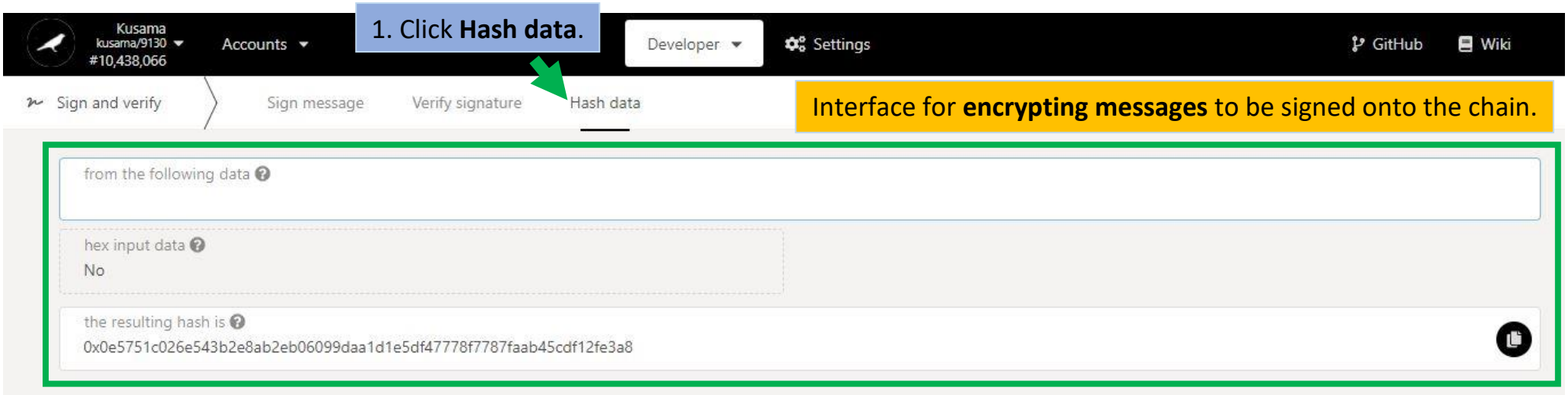
5. If any of the information provided is incorrect, the signature will be flagged as invalid!

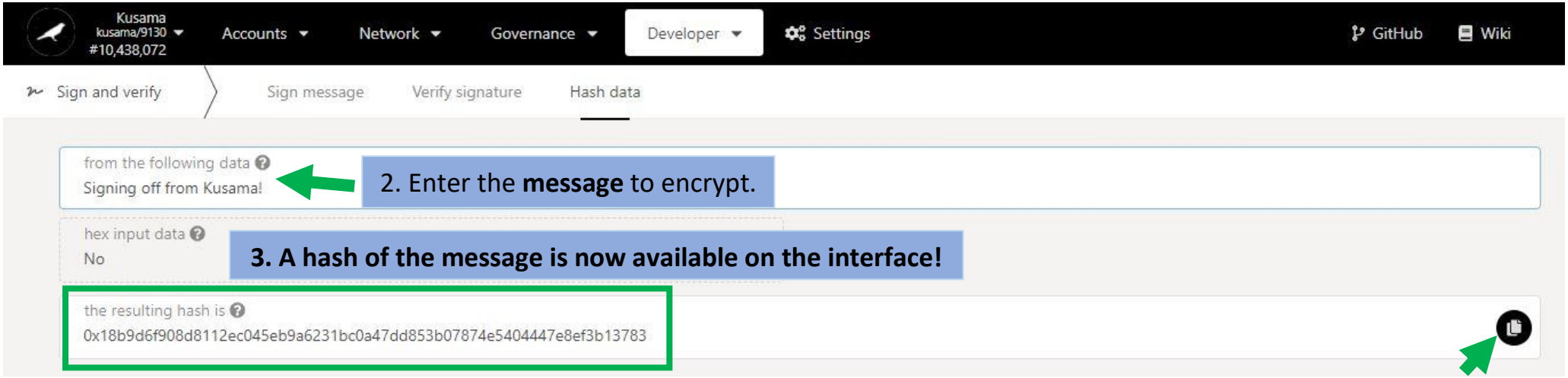


6. **Check and correct** the message that was signed.

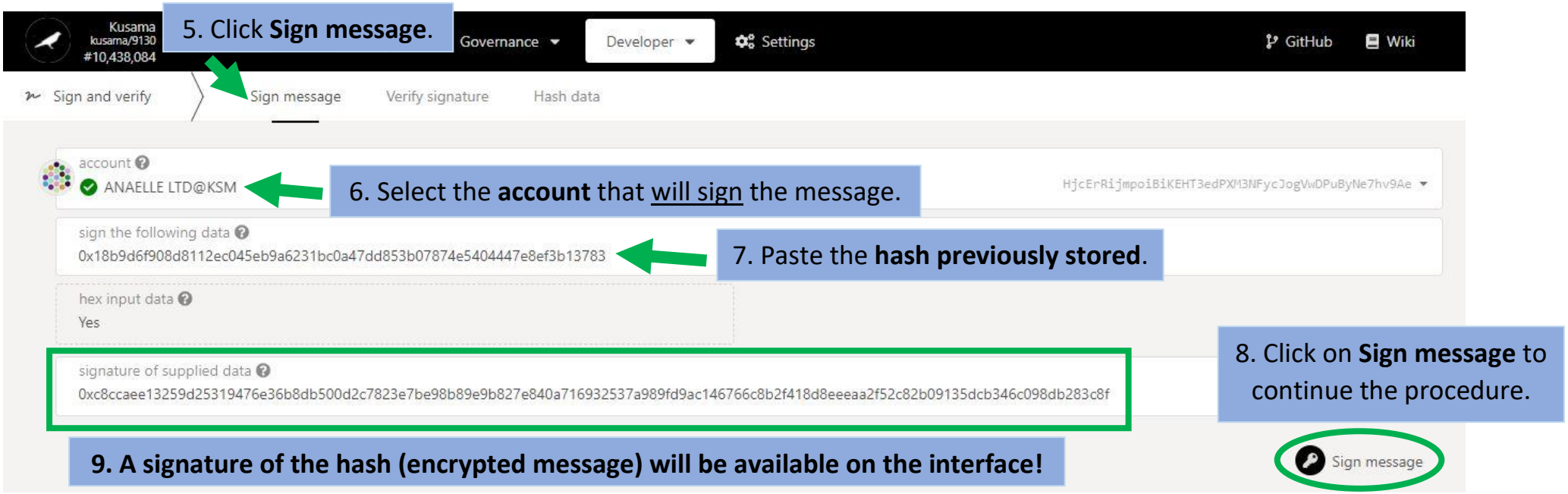
7. Once all the information provided is correct, the signature will be marked as valid!

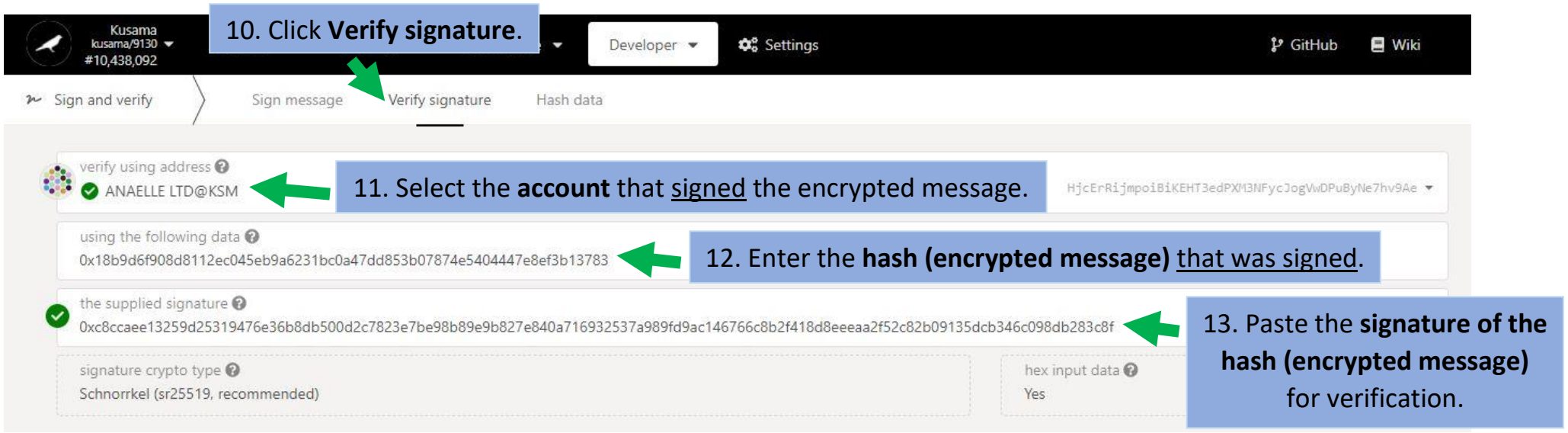
c) Hash data.



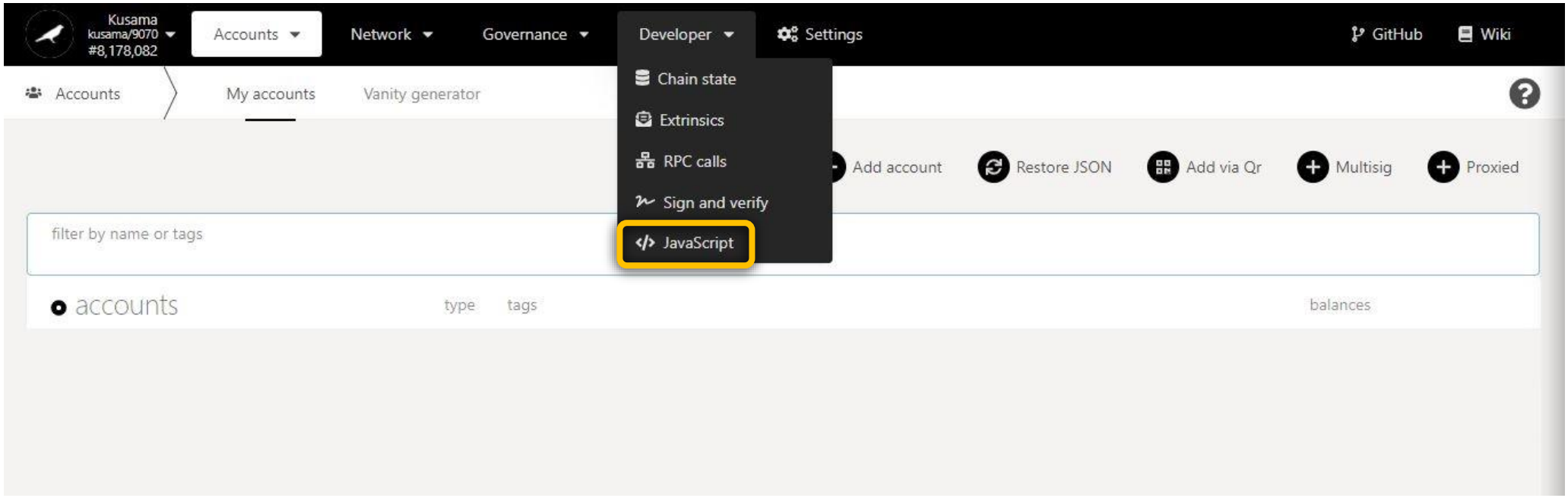


4. Copy the **hash and store it** for future use.


Kusama
kusama/9130
#10,438,084
5. Click **Sign message**.
Governance
Developer
Settings
GitHub
Wiki
Sign and verify
Sign message
Verify signature
Hash data
account
ANAELLE LTD@KSM
6. Select the **account** that will sign the message.
HjcErRijmpoiBiKEHT3edPXM3NFycJogVwDPuByNe7hv9Ae
sign the following data
0x18b9d6f908d8112ec045eb9a6231bc0a47dd853b07874e5404447e8ef3b13783
7. Paste the **hash previously stored**.
hex input data
Yes
signature of supplied data
0xc8ccaee13259d25319476e36b8db500d2c7823e7be98b89e9b827e840a716932537a989fd9ac146766c8b2f418d8eeeaa2f52c82b09135dcb346c098db283c8f
8. Click on **Sign message** to continue the procedure.
9. A signature of the hash (encrypted message) will be available on the interface!
Sign message


Kusama
kusama/9130
#10,438,092
10. Click **Verify signature**.
Developer
Settings
GitHub
Wiki
Sign and verify
Sign message
Verify signature
Hash data
verify using address
ANAELLE LTD@KSM
11. Select the **account** that signed the encrypted message.
HjcErRijmpoiBiKEHT3edPXM3NFycJogVwDPuByNe7hv9Ae
using the following data
0x18b9d6f908d8112ec045eb9a6231bc0a47dd853b07874e5404447e8ef3b13783
12. Enter the **hash (encrypted message)** that was signed.
the supplied signature
0xc8ccaee13259d25319476e36b8db500d2c7823e7be98b89e9b827e840a716932537a989fd9ac146766c8b2f418d8eeeaa2f52c82b09135dcb346c098db283c8f
13. Paste the **signature of the hash (encrypted message)** for verification.
signature crypto type
Schnorrkel (sr25519, recommended)
hex input data
Yes

5. **Javascript:** Interact with on-chain data through the console.



a) View and submit code snippets.

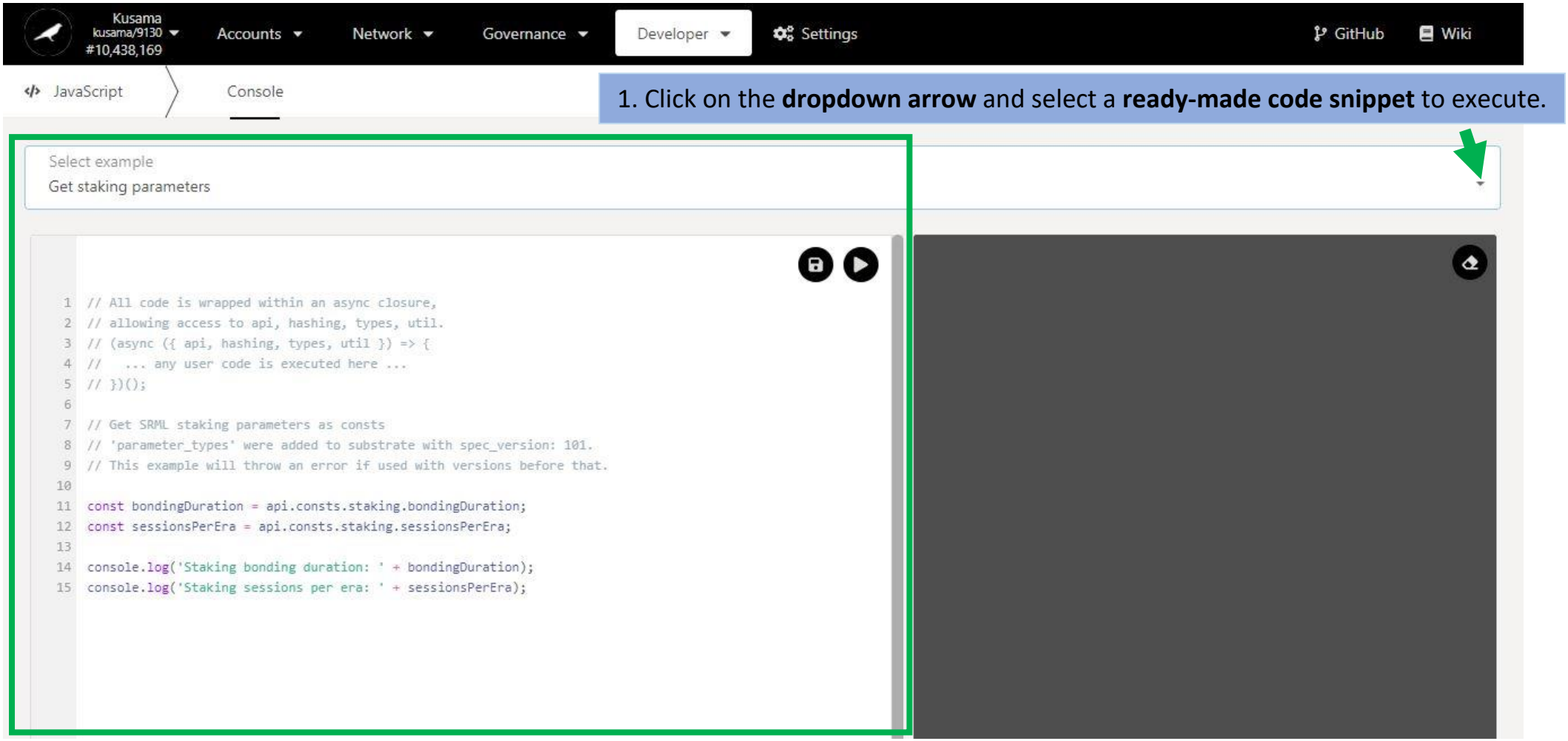
Kusama
kusama/9130
#10,438,101
Accounts
Network
Governance
Developer
Settings
GitHub
Wiki

JavaScript
Console

Interface for **executing JS scripts** on-chain in isolation.

Select example
Listen to new Head

```
// All code is wrapped within an async closure,
// allowing access to api, hashing, types, util.
// (async ({ api, hashing, types, util }) => {
//   ... any user code is executed here ...
// })();

// subscribe to new headers, printing the full info for 5 Blocks
let count = 0;
const unsub = await api.rpc.chain.subscribeNewHeads((header) => {
  console.log(`#${header.number}:`, header);

  if (++count === 5) {
    console.log('5 headers retrieved, unsubscribing');
    unsub();
  }
});
```


Kusama
kusama/9130
#10,438,169
Accounts
Network
Governance
Developer
Settings
GitHub
Wiki
JavaScript
Console
1. Click on the **dropdown arrow** and select a **ready-made code snippet** to execute.
Select example
Get staking parameters
// All code is wrapped within an async closure,
// allowing access to api, hashing, types, util.
// (async ({ api, hashing, types, util }) => {
// ... any user code is executed here ...
// })();
// Get SRML staking parameters as consts
// 'parameter_types' were added to substrate with spec_version: 101.
// This example will throw an error if used with versions before that.
const bondingDuration = api.consts.staking.bondingDuration;
const sessionsPerEra = api.consts.staking.sessionsPerEra;
console.log('Staking bonding duration: ' + bondingDuration);
console.log('Staking sessions per era: ' + sessionsPerEra);

Kusama
kusama/9130
#10,438,179

Accounts | Network | Governance | Developer | Settings | GitHub | Wiki

JavaScript | Console

Select example
Get staking parameters

```
// All code is wrapped within an async closure,
// allowing access to api, hashing, types, util.
// (async ({ api, hashing, types, util }) => {
//   ... any user code is executed here ...
// })();

// Get SRML staking parameters as consts
// 'parameter_types' were added to substrate with spec_version: 101.
// This example will throw an error if used with versions before that.

const bondingDuration = api.consts.staking.bondingDuration;
const sessionsPerEra = api.consts.staking.sessionsPerEra;

console.log('Staking bonding duration: ' + bondingDuration);
console.log('Number of sessions per era: ' + sessionsPerEra);
```

2. Change the **labels** (in green) for better legibility of the data.

Kusama
kusama/9130
#10,438,187

Accounts | Network | Governance | Developer | Settings | GitHub | Wiki

JavaScript | Console

Select example
Get staking parameters

3. Click on the **play button** to run the script.

```
// All code is wrapped within an async closure,
// allowing access to api, hashing, types, util.
// (async ({ api, hashing, types, util }) => {
//   ... any user code is executed here ...
// })();

// Get SRML staking parameters as consts
// 'parameter_types' were added to substrate with spec_version: 101.
// This example will throw an error if used with versions before that.

const bondingDuration = api.consts.staking.bondingDuration;
const sessionsPerEra = api.consts.staking.sessionsPerEra;

console.log('Staking bonding duration: ' + bondingDuration);
console.log('Number of sessions per era: ' + sessionsPerEra);
```

```
Staking bonding duration: 28
Number of sessions per era: 6
```

4. Results are now available in the JS console!

b) Customise code snippets.

JavaScript > Console

1. Click on the **dropdown arrow** and select a **ready-made code snippet** to customise.

Select example
Retrieve historic query data

```
// All code is wrapped within an async closure,
// allowing access to api, hashing, types, util.
// (async ({ api, hashing, types, util }) => {
//   ... any user code is executed here ...
// })();

// This example set shows how to make queries at a point
const ALICE = '5GrwvaEF5zXb26Fz9rcQpDWS57CtERHpNehXCPcNoHGKutQY';

// retrieve the balance, once-off at the latest block
const { data: { free } } = await api.query.system.account(ALICE);

console.log('Alice has a current balance of', free.toHuman());

// retrieve balance updates with an optional value callback
const balanceUnsub = await api.query.system.account(ALICE, ({ data: { free } }) => {
  console.log('Alice has an updated balance of', free.toHuman());
});

// retrieve the balance at a block hash in the past
const header = await api.rpc.chain.getHeader();
const prevHash = await api.rpc.chain.getBlockHash(header.number.unwrap().subn(42));
const { data: { free: prev } } = await api.query.system.account.at(prevHash, ALICE);

console.log('Alice had a balance of', prev.toHuman(), '(42 blocks ago)');

// useful in some situations - the value hash and storage entry size
const currHash = await api.query.system.account.hash(ALICE);
```

JavaScript Console

Select example
Retrieve historic query data

2. Click on the **play button** to run the script and show the results in the JS console.

```
// All code is wrapped within an async closure,
// allowing access to api, hashing, types, util.
// (async ({ api, hashing, types, util }) => {
//   ... any user code is executed here ...
// })();

// This example set shows how to make queries at a point
const ALICE = '5GrwvaEF5zXb26Fz9rcQpDWS57CtERHpNehXCPcNoHGKutQY';

// retrieve the balance, once-off at the latest block
const { data: { free } } = await api.query.system.account(ALICE);

console.log('Alice has a current balance of', free.toHuman());

// retrieve balance updates with an optional value callback
const balanceUnsub = await api.query.system.account(ALICE, ({ data: { free } }) => {
  console.log('Alice has an updated balance of', free.toHuman());
});

// retrieve the balance at a block hash in the past
const header = await api.rpc.chain.getHeader();
const prevHash = await api.rpc.chain.getBlockHash(header.number.unwrap().subn(42));
const { data: { free: prev } } = await api.query.system.account.at(prevHash, ALICE);

console.log('Alice had a balance of', prev.toHuman(), '(42 blocks ago)');

// useful in some situations - the value hash and storage entry size
const currHash = await api.query.system.account.hash(ALICE);
```

```
Alice has a current balance of 0
Alice has an updated balance of 0
Alice had a balance of 0 (42 blocks ago)
```

3. Click on the **eraser** button to clear the console.

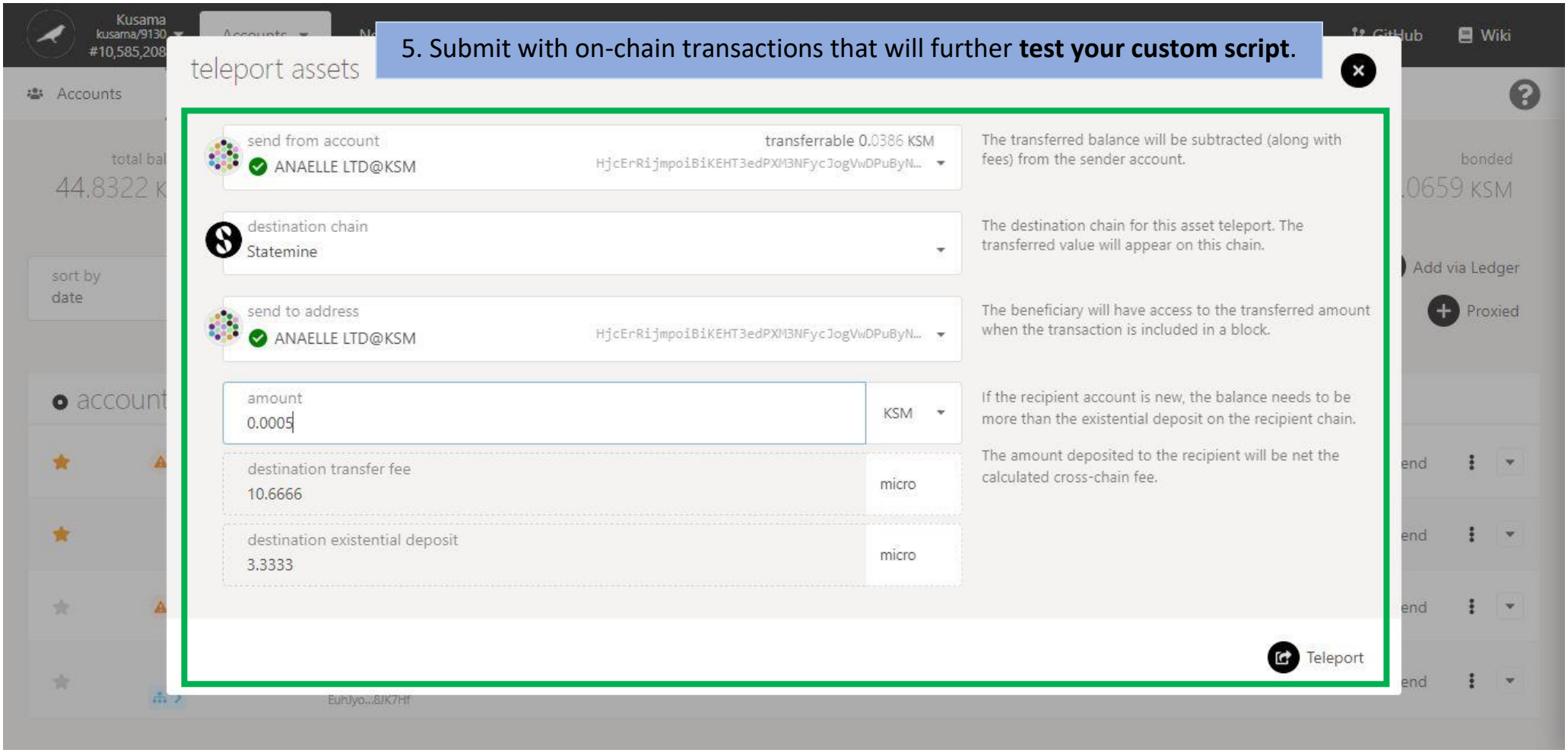
JavaScript Console

4. Change the **variables** (in blue) and the **labels** (in green) to customise the script, then run the script to show the results.

Select example
Retrieve historic query data

```
// All code is wrapped within an async closure,
// allowing access to api, hashing, types, util.
// (async ({ api, hashing, types, util }) => {
//   ... any user code is executed here ...
// })();

// This example set shows how to make queries at a point
const ANAELLELTD = 'HjcErRijmpoiBiKEHT3edPXM3NFycJogVwDPuByNe7hv9Ae';

// retrieve the balance, once-off at the latest block
const { data: { free } } = await api.query.system.account(ANAELLELTD);

console.log('AnaelleLTD has a current balance of', free.toHuman());

// retrieve balance updates with an optional value callback
const balanceUnsub = await api.query.system.account(ANAELLELTD, ({ data: { free } }) => {
  console.log('AnaelleLTD has an updated balance of', free.toHuman());
});

// retrieve the balance at a block hash in the past
const header = await api.rpc.chain.getHeader();
const prevHash = await api.rpc.chain.getBlockHash(header.number.unwrap().subn(600));
const { data: { free: prev } } = await api.query.system.account.at(prevHash, ANAELLELTD);

console.log('AnaelleLTD had a balance of', prev.toHuman(), '(600 blocks ago aka 1 hour ago)');
```

```
AnaelleLTD has a current balance of 5,038,674,509,552
AnaelleLTD has an updated balance of 5,038,674,509,552
AnaelleLTD had a balance of 5,038,674,509,552 (600 blocks ago aka 1 hour
```


5. Submit with on-chain transactions that will further **test your custom script**.
teleport assets
send from account
ANAELLE LTD@KSM
transferrable 0.0386 KSM
HjcErRijmpoiBiKEHT3edPXM3NFycJogVwDPuByN...
The transferred balance will be subtracted (along with fees) from the sender account.
destination chain
Statemine
The destination chain for this asset teleport. The transferred value will appear on this chain.
send to address
ANAELLE LTD@KSM
HjcErRijmpoiBiKEHT3edPXM3NFycJogVwDPuByN...
The beneficiary will have access to the transferred amount when the transaction is included in a block.
amount
0.0005
KSM
If the recipient account is new, the balance needs to be more than the existential deposit on the recipient chain.
destination transfer fee
10.6666
micro
The amount deposited to the recipient will be net the calculated cross-chain fee.
destination existential deposit
3.3333
micro
Teleport

JavaScript Console

Select example
Retrieve historic query data

6. Check the **updated results**.

```
// All code is wrapped within an async closure,
// allowing access to api, hashing, types, util.
// (async ({ api, hashing, types, util }) => {
//   ... any user code is executed here ...
// })();

// This example set shows how to make queries at a point
const ANAELLELTD = 'HjcErRijmpoiBiKEHT3edPXM3NFycJogVwDPuByNe7hv9Ae';

// retrieve the balance, once-off at the latest block
const { data: { free } } = await api.query.system.account(ANAELLELTD);

console.log('AnaelleLTD has a current balance of', free.toHuman());

// retrieve balance updates with an optional value callback
const balanceUnsub = await api.query.system.account(ANAELLELTD, ({ data: { free } }) => {
  console.log('AnaelleLTD has an updated balance of', free.toHuman());
});

// retrieve the balance at a block hash in the past
const header = await api.rpc.chain.getHeader();
const prevHash = await api.rpc.chain.getBlockHash(header.number.unwrap().subn(600));
const { data: { free: prev } } = await api.query.system.account.at(prevHash, ANAELLELTD);

console.log('AnaelleLTD had a balance of', prev.toHuman(), '(600 blocks ago aka 1 hour ago)');
```

```
AnaelleLTD has a current balance of 5,038,674,509,552
AnaelleLTD has an updated balance of 5,038,674,509,552
AnaelleLTD had a balance of 5,038,674,509,552 (600 blocks ago aka 1 hour
AnaelleLTD has an updated balance of 5,038,117,176,704
```

JavaScript Console

Select example
Retrieve historic query data

7. Change the **functions** (in purple) to further customise your script, then run the script to show the results.

```
// All code is wrapped within an async closure,
// allowing access to api, hashing, types, util.
// (async ({ api, hashing, types, util }) => {
//   ... any user code is executed here ...
// })();

// This example set shows how to make queries at a point
const ANAELLELTD = 'HjcErRijmpoiBiKEHT3edPXM3NFycJogVwDPuByNe7hv9Ae';

// retrieve the balance, once-off at the latest block
const { data: { free } } = await api.query.system.account(ANAELLELTD);

console.log('AnaelleLTD has a current balance of', free.toBigInt());

// retrieve balance updates with an optional value callback
const balanceUnsub = await api.query.system.account(ANAELLELTD, ({ data: { free } }) => {
  console.log('AnaelleLTD has an updated balance of', free.toHuman());
});

// retrieve the balance at a block hash in the past
const header = await api.rpc.chain.getHeader();
const prevHash = await api.rpc.chain.getBlockHash(header.number.unwrap().subn(600));
const { data: { free: prev } } = await api.query.system.account.at(prevHash, ANAELLELTD);

console.log('AnaelleLTD had a balance of', prev.toHuman(), '(600 blocks ago aka 1 hour ago)');
```

```
AnaelleLTD has a current balance of 5,038,674,509,552
AnaelleLTD has an updated balance of 5,038,674,509,552
AnaelleLTD had a balance of 5,038,674,509,552 (600 blocks ago aka 1 hour
AnaelleLTD has an updated balance of 5,038,117,176,704
AnaelleLTD has a current balance of 5038117176704
AnaelleLTD has an updated balance of 5,038,117,176,704
AnaelleLTD had a balance of 5,038,674,509,552 (600 blocks ago aka 1 hour
```

c) Back up code snippets.

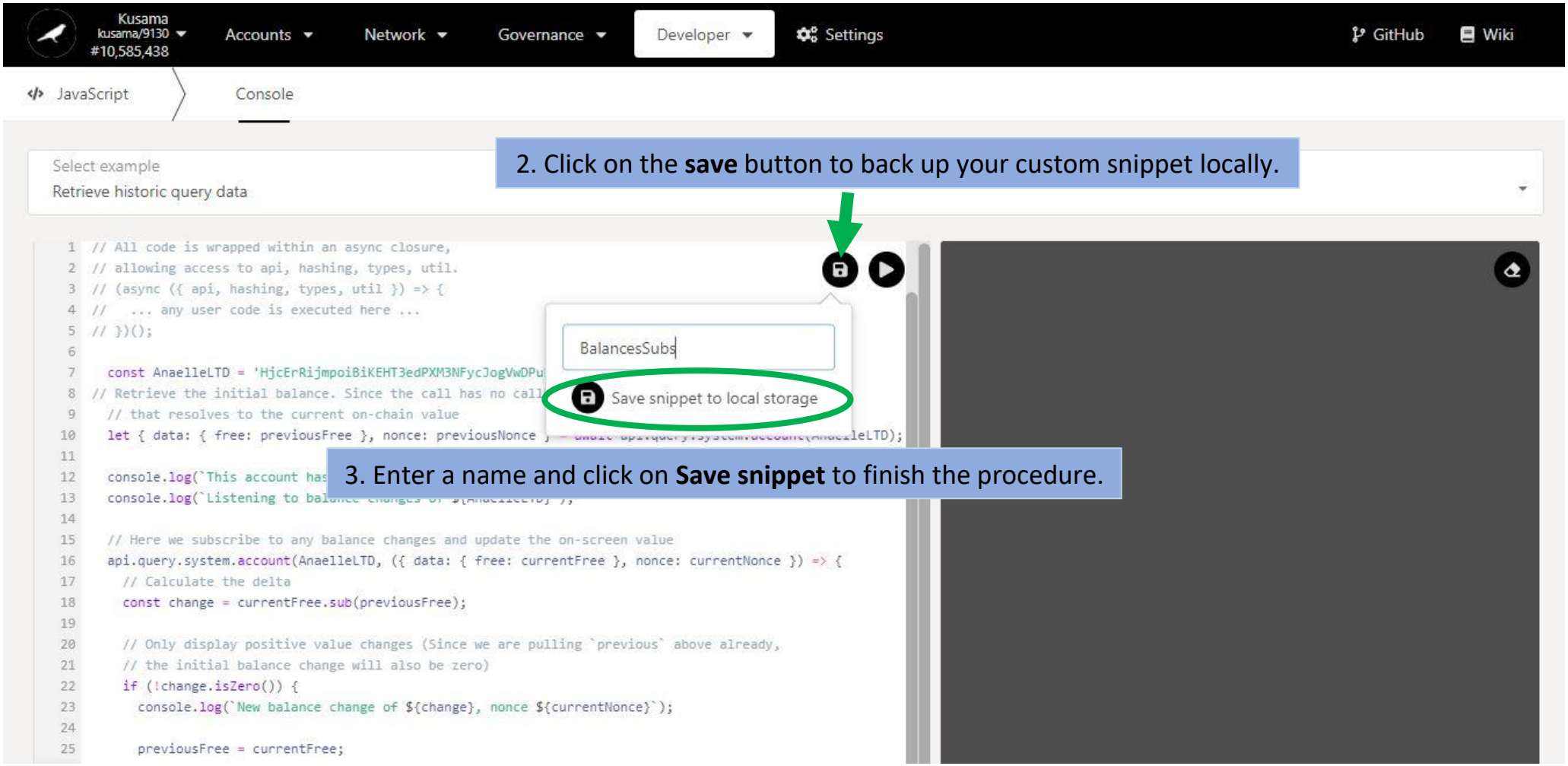
JavaScript | Console

Select example
Retrieve historic query data

1. Add more **custom-made code** to your script, then run the script to show the results, and clear the console.

```javascript
// All code is wrapped within an async closure,
// allowing access to api, hashing, types, util.
// (async ({ api, hashing, types, util }) => {
//   ... any user code is executed here ...
// })();

  const AnaelleLTD = 'HjcErRijmpoiBiKEHT3edPXM3NFycJogVwDPuByNe7hv9Ae';
// Retrieve the initial balance. Since the call has no callback, it is simply a promise
  // that resolves to the current on-chain value
  let { data: { free: previousFree }, nonce: previousNonce } = await api.query.system.account(AnaelleLTD);

  console.log(`This account has a balance of ${previousFree}, nonce ${previousNonce}`);
  console.log(`Listening to balance changes of ${AnaelleLTD}`);

  // Here we subscribe to any balance changes and update the on-screen value
  api.query.system.account(AnaelleLTD, ({ data: { free: currentFree }, nonce: currentNonce }) => {
    // Calculate the delta
    const change = currentFree.sub(previousFree);

    // Only display positive value changes (Since we are pulling `previous` above already,
    // the initial balance change will also be zero)
    if (!change.isZero()) {
      console.log(`New balance change of ${change}, nonce ${currentNonce}`);

      previousFree = currentFree;
      previousNonce = currentNonce;
    }
  });
```

```
This account has a balance of 5038117176704, nonce 220
Listening to balance changes of HjcErRijmpoiBiKEHT3edPXM3NFycJogVwDPuBy
```


Kusama
kusama/9130
#10,585,438
Accounts
Network
Governance
Developer
Settings
GitHub
Wiki
JavaScript
Console
Select example
Retrieve historic query data
2. Click on the **save** button to back up your custom snippet locally.
BalancesSubs
Save snippet to local storage
3. Enter a name and click on **Save snippet** to finish the procedure.
// All code is wrapped within an async closure,
// allowing access to api, hashing, types, util.
// (async ({ api, hashing, types, util }) => {
// ... any user code is executed here ...
// })();
const AnaelleLTD = 'HjcErRijmpoiBiKEHT3edPXM3NFycJogVwDPu
// Retrieve the initial balance. Since the call has no call
// that resolves to the current on-chain value
let { data: { free: previousFree }, nonce: previousNonce
console.log(`This account has
console.log(`Listening to bal
// Here we subscribe to any balance changes and update the on-screen value
api.query.system.account(AnaelleLTD, ({ data: { free: currentFree }, nonce: currentNonce }) => {
// Calculate the delta
const change = currentFree.sub(previousFree);
// Only display positive value changes (Since we are pulling `previous` above already,
// the initial balance change will also be zero)
if (!change.isZero()) {
console.log(`New balance change of ${change}, nonce ${currentNonce}`);
previousFree = currentFree;

Kusama
kusama/9130
#10,585,442
Accounts
Network
Governance
Developer
Settings
GitHub
Wiki

4. Your snippet has been renamed!

Select example
BalancesSubs

5. Click on the **dropdown arrow** to check the list of available snippets.

```
// All code is wrapped within an async closure,
// allowing access to api, hashing, types, util.
// (async ({ api, hashing, types, util }) => {
//   ... any user code is executed here ...
// })();

  const AnaelleLTD = 'HjcErRijmpoiBiKEHT3edPXM3NFycJogVwDPuByNe7hv9Ae';
// Retrieve the initial balance. Since the call has no callback, it is simply a promise
  // that resolves to the current on-chain value
  let { data: { free: previousFree }, nonce: previousNonce } = await api.query.system.account(AnaelleLTD);

  console.log(`This account has a balance of ${previousFree}, nonce ${previousNonce}`);
  console.log(`Listening to balance changes of ${AnaelleLTD}`);

  // Here we subscribe to any balance changes and update the on-screen value
  api.query.system.account(AnaelleLTD, ({ data: { free: currentFree }, nonce: currentNonce }) => {
    // Calculate the delta
    const change = currentFree.sub(previousFree);

    // Only display positive value changes (Since we are pulling `previous` above already,
    // the initial balance change will also be zero)
    if (!change.isZero()) {
      console.log(`New balance change of ${change}, nonce ${currentNonce}`);

      previousFree = currentFree;
```

Kusama
kusama/9130
#10,585,544

Accounts | Network | Governance | Developer | Settings | GitHub | Wiki

JavaScript | Console

6. Your snippet is now (temporarily) stored in the dropdown menu!

BalancesSubs Custom

Get authoring information RPC

Listen to new Head RPC

Get state metadata RPC

Get system information RPC

```
console.log(`This account has a balance of ${previousFree}, nonce ${previousNonce}`);
console.log(`Listening to balance changes of ${AnaelleLTD}`);

// Here we subscribe to any balance changes and update the on-screen value
api.query.system.account(AnaelleLTD, ({ data: { free: currentFree }, nonce: currentNonce }) => {
  // Calculate the delta
  const change = currentFree.sub(previousFree);

  // Only display positive value changes (Since we are pulling `previous` above already,
  // the initial balance change will also be zero)
  if (!change.isZero()) {
    console.log(`New balance change of ${change}, nonce ${currentNonce}`);

    previousFree = currentFree;
```