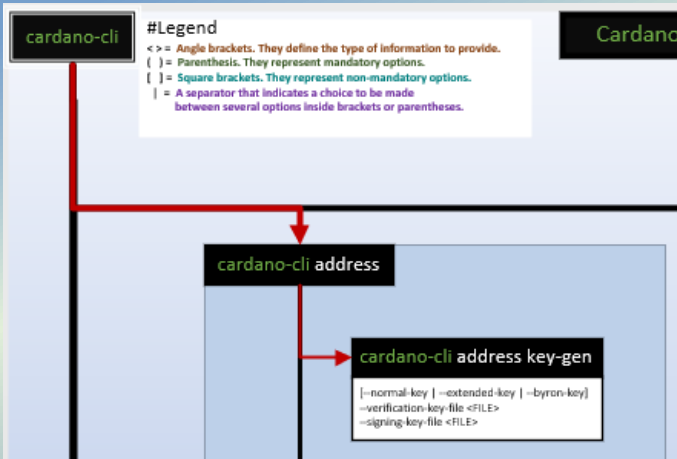


Questo documento ha lo scopo di spiegare dettagliatamente come interpretare i comandi cardano-cli e le relative opzioni al fine di essere in grado di assemblarli autonomamente, se necessario. Per fare ciò, è necessario disporre di un computer e installare un Nodo della blockchain di Cardano e l'interfaccia a linea di comando di Cardano (cardano-cli). Inizierai con comandi semplici e aumenterai gradualmente la complessità man mano che il tutorial avanza.

Primo esercizio: Creazione delle chiavi di pagamento e stake

Air Gap

1 Prima di tutto, individua il comando che utilizzerai per le tue chiavi di pagamento.



2 Hai 5 opzioni in totale.

Le prime 3 opzioni sono racchiuse tra parentesi quadre con 2 separatori, il che indica che la scelta tra queste 3 opzioni non è obbligatoria, poiché verrà utilizzata una chiave normale per impostazione predefinita (default) se non viene specificato nulla. Per questo esempio, non le utilizzerai.

cardano-cli address key-gen

```
[ -normal-key | -extended-key | -byron-key ]
--verification-key-file <FILE>
--signing-key-file <FILE>
```

3 Le seguenti 2 opzioni dovranno essere usate.

Le parentesi acute indicano il tipo di informazione <File>. Devi fornire il nome che darai ai tuoi 2 file di chiave.

cardano-cli address key-gen

```
[ -normal-key | -extended-key | -byron-key ]
--verification-key-file payment.vkey
--signing-key-file payment.skey
```

4 Questo è il risultato finale di questo semplice comando sul tuo terminale.

```
user@computer:~$ cardano-cli address key-gen \
> --verification-key-file payment.vkey \
> --signing-key-file payment.skey
```

5 Ora che le tue chiavi di pagamento sono pronte, devi creare le tue chiavi di stake. Questa è ancora più facile perché c'è solo un tipo di chiave di stake.



cardano-cli stake-address key-gen

```
--verification-key-file <FILE>
--signing-key-file <FILE>
```

6 Come il **3** Devi indicare il tipo di informazione. <File>

cardano-cli stake-address key-gen

```
--verification-key-file stake.vkey
--signing-key-file stake.skey
```

7 E come risultato finale sul terminale...

```
user@computer:~$ cardano-cli stake-address key-gen \
> --verification-key-file stake.vkey \
> --signing-key-file stake.skey
```

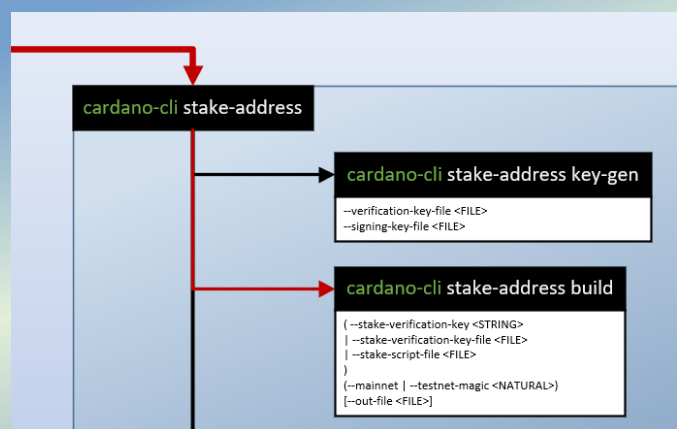
Attenzione. Si consiglia di generare e utilizzare le tue chiavi di pagamento e di stake per firmare transazioni in un ambiente "Air Gap" per motivi di sicurezza. <https://developers.cardano.org/docs/get-started/air-gap>

Ora che le tue 2 coppie di chiavi sono state create, sarai in grado di creare un indirizzo di stake che ti permetterà di verificare l'importo delle tue ricompense e di prelevarle quando utilizzati in una transazione con la tua stake.skey.

Secondo esercizio: Creazione di un indirizzo di stake

Air Gap

1 Individua il comando che utilizzerai per creare il tuo indirizzo di stake.



2 Hai un totale di 6 opzioni.

Le prime 3 opzioni sono racchiuse tra parentesi tonde e hanno 2 separatori, il che indica che è obbligatorio fare una scelta tra queste tre opzioni.

cardano-cli stake-address build

```
( --stake-verification-key <STRING>
| --stake-verification-key-file <FILE>
| --stake-script-file <FILE>
)
(--mainnet | --testnet-magic <NATURAL>)
[--out-file <FILE>]
```

3 Userai l'opzione stake-verification-key-file.

Le parentesi acute indicano il tipo di informazione <File>. Questa volta, devi fornire il percorso che punta alla tua stake.vkey.

cardano-cli stake-address build

```
( stake-verification-key <STRING>
| --stake-verification-key-file stake.vkey
| stake-script-file <FILE>
)
(--mainnet | --testnet-magic <NATURAL>)
[--out-file <FILE>]
```

4 Ora hai 2 opzioni tra parentesi tonde.

Dovrai specificare la rete utilizzata e, se si tratta della testnet, menzionare il numero magico della rete. In questo caso utilizzeremo la mainnet.

cardano-cli stake-address build

```
--stake-verification-key-file stake.vkey
--mainnet | testnet-magic <NATURAL>
[--out-file <FILE>]
```

5 E l'ultima opzione --out-file <FILE>

Questa opzione non è obbligatoria perché è racchiusa tra parentesi quadre. Tuttavia, se non utilizzi questa opzione, l'output (indirizzo di stake) del comando verrà visualizzato sul tuo terminale anziché salvato in un file. Poiché avrai bisogno di utilizzare questo indirizzo di stake successivamente, diamo un nome ad esso e salviamolo in un file.

cardano-cli stake-address build

```
--stake-verification-key-file stake.vkey
--mainnet
--out-file stake.addr
```

6 Questo è come il comando viene visualizzato sul tuo terminale.

```
user@computer:~$ cardano-cli stake-address build \
> --stake-verification-key-file stake.vkey \
> --mainnet \
> --out-file stake.addr
```

7 Ecco cosa dovresti avere finora.

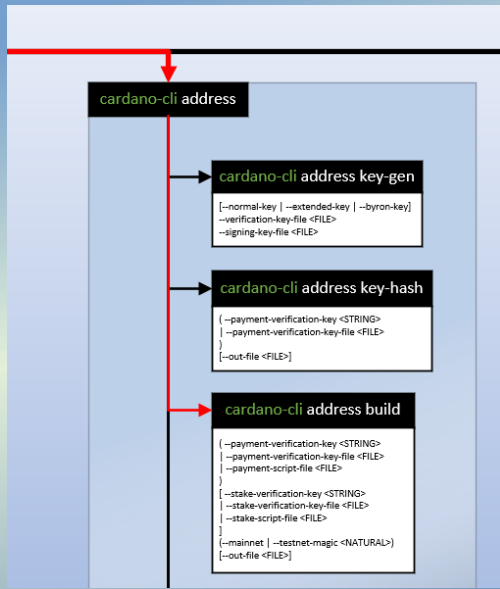
```
user@computer:~$ ls
payment.vkey  payment.skey  stake.vkey
stake.skey   stake.addr
```

Ora che le tue due coppie di chiavi e il tuo indirizzo di stake sono stati creati, sarai in grado di creare un indirizzo combinando la tua chiave di pagamento con la chiave di stake in modo che il saldo nell'indirizzo generato siano inclusi nel protocollo di stake con le tue ricompense.

Terzo esercizio: Creazione di un pagamento con un file di indirizzo di stake

Air Gap

1 Individua il comando che utilizzerai per il tuo pagamento con il file di indirizzo di stake.



2 Hai 9 opzioni in totale

È possibile creare un indirizzo di pagamento utilizzando solo la tua chiave di pagamento senza collegarla alla tua chiave di stake, il che spiega perché il primo gruppo di opzioni è obbligatorio e non il secondo gruppo.

```
cardano-cli address build
(
  --payment-verification-key <STRING>
  | --payment-verification-key-file <FILE>
  | --payment-script-file <FILE>
)
[
  --stake-verification-key <STRING>
  | --stake-verification-key-file <FILE>
  | --stake-script-file <FILE>
]
(--mainnet | --testnet-magic <NATURAL>)
[--out-file <FILE>]
```

3 Utilizziamo entrambe le chiavi verifica di pagamento e di stake.

Di nuovo, secondo le parentesi acute <FILE>, devi definire il percorso ai file di payment.vkey e di stake.vkey.

```
cardano-cli address build
(
  --payment-verification-key <STRING>
  | --payment-verification-key-file payment.vkey
  | --payment-script-file <FILE>
)
[
  --stake-verification-key <STRING>
  | --stake-verification-key-file stake.vkey
  | --stake-script-file <FILE>
]
(--mainnet | --testnet-magic <NATURAL>)
[--out-file <FILE>]
```

4 Ora dovrai specificare la rete da utilizzare e il nome del file per il tuo indirizzo.

```
cardano-cli address build
--payment-verification-key-file payment.vkey
--stake-verification-key-file stake.vkey
--mainnet | --testnet-magic <NATURAL>
--out-file paymentwithstake.addr
```

5 Questo è il risultato finale.

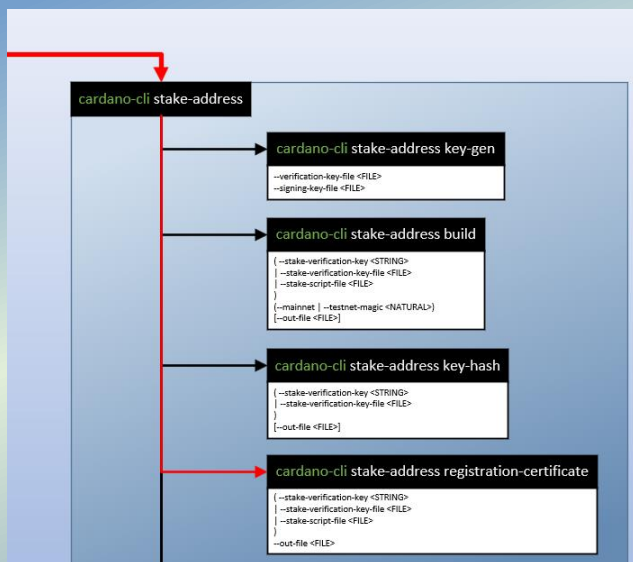
```
user@computer:~$ cardano-cli address build \
> --payment-verification-key-file payment.vkey \
> --stake-verification-key-file stake.vkey \
> --mainnet \
> --out-file paymentwithstake.addr
```

Puoi copiare il contenuto di paymentwithstake.addr in un editor di testo e incollarlo in una transazione del wallet Cardano che usi di solito e inviare alcuni ADA ad esso. (10 ADA dovrebbero essere sufficienti per iniziare)

Quarto esercizio: Creazione di un certificato di stake

Air Gap

1 Individua il comando che andrai ad usare per il tuo certificato di stake.



2 Hai 4 opzioni

Per poter partecipare al protocollo e mettere i tuoi ADA in stake, devi collegare la tua chiave di verifica dello stake a un certificato che invierai alla blockchain nei prossimi esercizi. Il comando per creare il tuo certificato è abbastanza semplice. Devi solo fornire una di queste 3 opzioni obbligatorie e specificare il nome del file che fungerà da certificato.

```
cardano-cli stake-address registration-certificate
(
  --stake-verification-key <STRING>
  | --stake-verification-key-file <FILE>
  | --stake-script-file <FILE>
)
--out-file <FILE>

cardano-cli stake-address registration-certificate
(
  --stake-verification-key <STRING>
  | --stake-verification-key-file stake.vrf
  | --stake-script-file <FILE>
)
--out-file stake.cert
```

3 Questo è il risultato finale di come il comando dovrebbe apparire sul tuo terminale.

```
user@computer:~$ cardano-cli stake-address registration-certificate \
> --stake-verification-key-file stake.vkey \
> --out-file stake.cert
```

4 Ecco cosa dovresti avere finora.

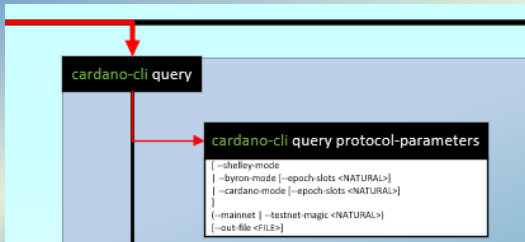
```
user@computer:~$ ls
payment.vkey    payment.skey    stake.vkey      stake.skey
stake.addr      paymentwithstake.addr  stake.cert
```

Ora otterrai i parametri del protocollo e l'indicatore di avanzamento della blockchain (tip) in modo da poter iniziare a costruire la tua primissima transazione.

Quinto esercizio: Ottenere i parametri di protocollo

Hot Node

1 Prima di tutto, per la tua transazione, avrai bisogno dei parametri di protocollo per il calcolo delle commissioni.



2 Hai 6 opzioni e 2 sotto-opzioni in totale

```
cardano-cli query protocol-parameters
[
  --shelley-mode
  | --byron-mode [--epoch-slots <NATURAL>]
  | --cardano-mode [--epoch-slots <NATURAL>]
]
(--mainnet | --testnet-magic <NATURAL>)
[--out-file <FILE>]
```

3 Salta le opzioni "mode". Menziona la rete desiderata e il nome del file da creare.

```
cardano-cli query protocol-parameters
[
  --shelley-mode
  | --byron-mode [--epoch-slots <NATURAL>]
  | --cardano-mode [--epoch-slots <NATURAL>]
]
--mainnet | --testnet-magic <NATURAL>
--out-file protocol.json
```

4 Questo è il risultato finale di come questo comando apparirà sul tuo terminale.

```
user@computer:~$ cardano-cli query protocol-parameters \
> --mainnet \
> --out-file protocol.json
```

5 Ora procedi e vediamo contenuto di quel file:

```
user@computer:~$ cat protocol.json
```

Nel file protocol.json cercherai il deposito da effettuare sulla blockchain per registrare il tuo indirizzo di stake e partecipare al protocollo di staking. Questo deposito può essere recuperato in qualsiasi momento se deregistri il tuo indirizzo.

6 Prendi nota dell'importo del deposito, poiché ne avrai bisogno in seguito. L'importo è espresso in Lovelace. (1 ADA = 1.000.000 Lovelace)

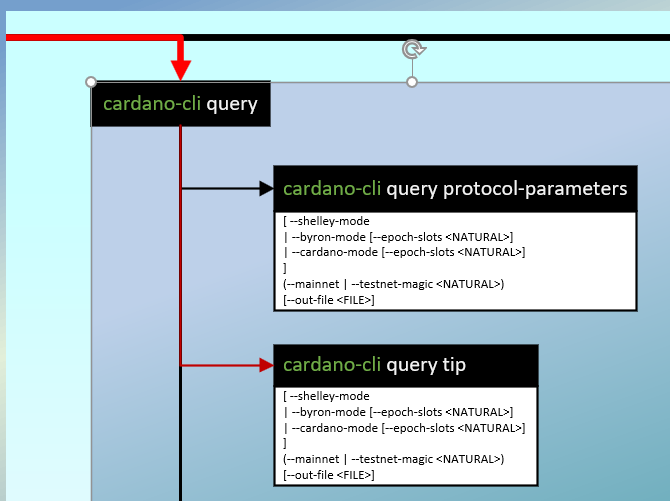
```
"poolRetireMaxEpoch": 18,
"protocolVersion": {
  "major": 8,
  "minor": 0
},
"stakeAddressDeposit": 2000000,
"stakePoolDeposit": 500000000,
"stakePoolTargetNum": 500,
"treasuryCut": 0.2,
"txFeeFixed": 155381,
"txFeePerByte": 44,
"utxoCostPerByte": 4310,
"utxoCostPerWord": null
```

7 Ora devi prendere il tuo file protocol.json e trasferirlo nel tuo ambiente "Air Gap" per poter calcolare le commissioni quando costruirai le tue transazioni.

! Con l'implementazione del CIP-1694 e l'avvicinarsi dell'era di Voltaire, sarà possibile per i possessori di ADA della comunità, con l'aiuto del comitato costituzionale e dei Dreps, modificare i parametri del protocollo attraverso un sistema di voto ben strutturato. Per questo motivo, è importante assicurarti di avere le modifiche più recenti di questi protocolli nel tuo ambiente "Air Gap", poiché ciò potrebbe avere un impatto diretto sui vari parametri che circondano le tue transazioni.

Sesto esercizio: Ottiene la punta (tip) attuale del nodo Hot Node

1 Nel prossimo esercizio avrai bisogno di conoscere la punta (tip) attuale del nodo per calcolare il nostro TTL. (La descrizione dettagliata sarà fornita nel prossimo esercizio)



2 Come nell'esercizio precedente, avrai a disposizione 6 opzioni e 2 sotto-opzioni.

Ora che stai iniziando a comprendere appieno il principio, puoi saltare alcuni passaggi. Non è necessario creare un file, hai solo bisogno del numero dello slot.

```
cardano-cli query tip
[ -shelley-mode
- byron-mode [-epoch-slots <NATURAL>]
- cardano-mode [-epoch-slots <NATURAL>]
]
[ -mainnet | -testnet-magic <NATURAL> ]
[ -out-file <FILE> ]
```

3 Questo è il risultato finale sul tuo terminale.

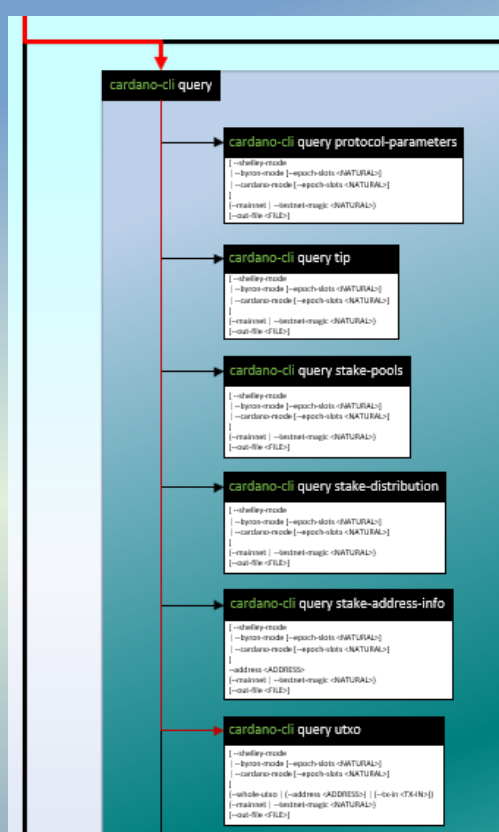
```
user@computer:~$ cardano-cli query tip \
> --mainnet
```

4 Notare il notare di slot

```
{
  "block": 8749125,
  "epoch": 410,
  "era": "Babbage",
  "hash": "503e4af96abc18e1d4b5de08e0d35cb508e364...",
  "slot": 92027764,
  "syncProgress": "100.00"
}
```

Settimo esercizio: Interrogare gli UTXO Hot Node

1 Ora interrogherai gli UTXO dell'indirizzo paymentwithstake.addr. (Se gli hai inviato degli ADA)



2 Questo comando ha 9 opzioni e 2 sotto-opzioni.

Dovrai consumare almeno un UTXO come input per la tua transazione. Una transazione può contenere diversi input e diversi output, ma in questo caso dovresti avere solo un UTXO associato al tuo indirizzo paymentwithstake.addr perché hai effettuato solo un deposito di 10 ADA su questo indirizzo. Quindi utilizzeremo solo ciò che è obbligatorio. In breve, utilizzeremo il tuo paymentwithstake.addr, la rete da utilizzare e successivamente creeremo un file per trasferire questa lista di UTXO nel tuo ambiente "air gap".

```
cardano-cli query utxo
[ -shelley-mode
- byron-mode [-epoch-slots <NATURAL>]
- cardano-mode [-epoch-slots <NATURAL>]
]
[ -whole-utxo | -address <ADDRESS> | (-tx-in <TX-IN>) ]
[ -mainnet | -testnet-magic <NATURAL> ]
[ -out-file <FILE> ]
```

3 Questo è come dovrebbe apparire sul tuo terminale.

```
cardano-cli query utxo
[ -shelley-mode
- byron-mode [-epoch-slots <NATURAL>]
- cardano-mode [-epoch-slots <NATURAL>]
]
[ -whole-utxo | -address paymentwithstake.addr | (-tx-in <TX-IN>) ]
[ -mainnet | -testnet-magic <NATURAL> ]
[ -out-file UTXO.addrs ]
```

```
user@computer:~$ cardano-cli query utxo \
> --address paymentwithstake.addr
> --mainnet
> --out-file utxo.addrs
```

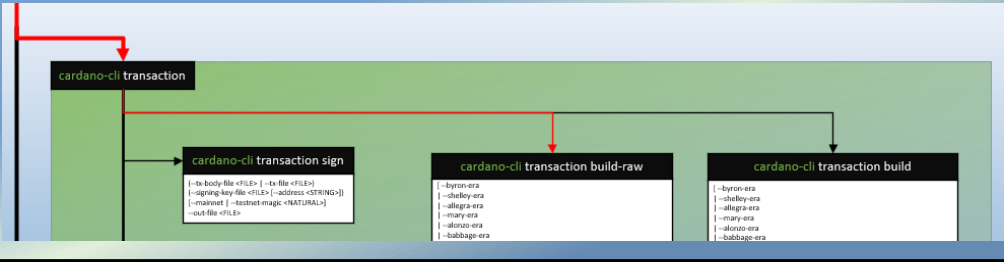
4 Il contenuto del file utxo.addr dovrebbe apparire come segue. L'UTXO del tuo deposito di 10 ADA corrisponde a 10.000.000 Lovelace. Ora puoi prendere questo file e inserirlo nel tuo ambiente "Air Gap". Ne avrai bisogno a breve.

TxHash	TxIx	Amount
1234a4d18e9dkhb34234kjbvdec3ad81e299c1a523443453561e61ce9bf8608e8c802df3b7f8c	0	10000000 lovelace + TxOutDatumNone

È ora il momento di creare la tua prima transazione, che verrà utilizzata per inviare il tuo certificato di stake. Prima di iniziare, ciò che vedrai potrebbe sembrare intimidatorio, ma seguendo passo dopo passo dovresti essere in grado di capire perché e come ridurre le prossime opzioni a un totale di 6 opzioni per il processo di transazione. Per motivi di sicurezza, in questo tutorial, utilizzerai metodi che coinvolgono il comando "cardano-cli transaction build-raw" invece del comando "cardano-cli transaction build", poiché può essere creato in un ambiente offline.

Ottavo esercizio: Creazione della bozza della tua prima transazione Air Gap

1 Trova il comando "cardano-cli transaction build-raw"



2 Iniziamo gradualmente dall'alto verso il basso.

```
cardano-cli transaction build-raw
[
  --byron-era
  | --shelley-era
  | --allegra-era
  | --mary-era
  | --alonzo-era
  | --babbage-era
]
```

3 Le prime 5 opzioni sono facoltative e racchiuse tra [parentesi quadre]. Se non viene specificato nulla, di default verrà utilizzata l'era di Mary.

4 La tua transazione non coinvolge uno script, quindi puoi saltare le prossime 2 opzioni.

5 Per le prossime opzioni e le sue 20 sotto-opzioni, è necessaria una spiegazione.

All'interno di una parentesi di opzione possono essere presenti sotto-opzioni definite da colonne. È per questo che esiste una nozione di priorità e un ordine particolare da rispettare quando si costruisce una transazione. In questo caso, si sa che ci sono 3 colonne distinte che definiscono l'ordine con cui le opzioni devono essere inserite se vogliamo che il corpo (body) della transazione venga prodotto correttamente.

```
cardano-cli transaction build-raw
[
  --byron-era
  | --shelley-era
  | --allegra-era
  | --mary-era
  | --alonzo-era
  | --babbage-era
]
[--script-valid | --script-invalid]
```

```
cardano-cli transaction build-raw
[
  --byron-era
  | --shelley-era
  | --allegra-era
  | --mary-era
  | --alonzo-era
  | --babbage-era
]
[
  --script-valid | --script-invalid
]
```

```
1 [ --script-valid | --script-invalid ]
  [ --tx-in <TX-IN>
2 [ --spending-tx-in-reference <TX-IN>
  | --spending-plutus-script-v2
3 [ --spending-reference-tx-in-datum-cbor-file
  | --spending-reference-tx-in-datum-file
```

6 Prenditi il tempo necessario per analizzare attentamente l'ordine di priorità dell'opzione "tx-in" e le relative parentesi.

7 "--tx-in" è obbligatorio, ma non le sue sotto-opzioni

8 Per quanto segue:

```
[
  --script-valid | --script-invalid
]
[
  --tx-in <TX-IN>
  [
    --spending-tx-in-reference <TX-IN>
    --spending-plutus-script-v2
    (
      --spending-reference-tx-in-datum-cbor-file <CBOR FILE>
      | --spending-reference-tx-in-datum-file <JSON FILE>
      | --spending-reference-tx-in-datum-value <JSON VALUE>
      | --spending-reference-tx-in-inline-datum-present
    )
    (
      --spending-reference-tx-in-redeemer-cbor-file <CBOR FILE>
      | --spending-reference-tx-in-redeemer-file <JSON FILE>
      | --spending-reference-tx-in-redeemer-value <JSON VALUE>
    )
    --spending-reference-tx-in-execution-units <INT, INT>
    --simple-script-tx-in-reference <TX-IN>
    | --tx-in-script-file <FILE>
  ]
  [
    --tx-in-datum-cbor-file <CBOR FILE>
    | --tx-in-datum-file <JSON FILE>
    | --tx-in-datum-value <JSON VALUE>
    | --tx-in-inline-datum-present
  ]
  (
    --tx-in-redeemer-cbor-file <CBOR FILE>
    | --tx-in-redeemer-file <JSON FILE>
    | --tx-in-redeemer-value <JSON VALUE>
  )
  --tx-in-execution-units <INT, INT>
]
[--read-only-tx-in-reference <TX-IN>]
[--tx-in-collateral <TX-IN>]
```

```
[
  --tx-in <TX-IN>
  [
    --spending-tx-in-reference <TX-IN>
    --spending-plutus-script-v2
    (
      --spending-reference-tx-in-datum-cbor-file <CBOR FILE>
      | --spending-reference-tx-in-datum-file <JSON FILE>
      | --spending-reference-tx-in-datum-value <JSON VALUE>
      | --spending-reference-tx-in-inline-datum-present
    )
    (
      --spending-reference-tx-in-redeemer-cbor-file <CBOR FILE>
      | --spending-reference-tx-in-redeemer-file <JSON FILE>
      | --spending-reference-tx-in-redeemer-value <JSON VALUE>
    )
    --spending-reference-tx-in-execution-units <INT, INT>
    --simple-script-tx-in-reference <TX-IN>
    | --tx-in-script-file <FILE>
  ]
  [
    --tx-in-datum-cbor-file <CBOR FILE>
    | --tx-in-datum-file <JSON FILE>
    | --tx-in-datum-value <JSON VALUE>
    | --tx-in-inline-datum-present
  ]
  (
    --tx-in-redeemer-cbor-file <CBOR FILE>
    | --tx-in-redeemer-file <JSON FILE>
    | --tx-in-redeemer-value <JSON VALUE>
  )
  --tx-in-execution-units <INT, INT>
]
]]
```

```
[
  --read-only-tx-in-reference <TX-IN>
  | --tx-in-collateral <TX-IN>
  | --tx-out-return-collateral <ADDRESS VALUE>
  | --tx-total-collateral <INTEGER>
  | --required-signer <FILE> | --required-signer-hash <HASH>
  | --tx-out <ADDRESS VALUE>
  [
    --tx-out-datum-hash <HASH>
    | --tx-out-datum-hash-cbor-file <CBOR FILE>
  ]
]
```

9 Circa "--required-signer-hash <HASH>"

Questa opzione non sarà utile in questo momento per la tua transazione, che sarà utilizzata per inviare il tuo stake.cert sulla blockchain, ma è importante notare che sarà molto utile nell'esercizio relativo al voto di governo.

```
[
  --read-only-tx-in-reference <TX-IN>
  | --tx-in-collateral <TX-IN>
  | --tx-out-return-collateral <ADDRESS VALUE>
  | --tx-total-collateral <INTEGER>
  | --required-signer <FILE> | --required-signer-hash <HASH>
  | --tx-out <ADDRESS VALUE>
]
```

10 Finalmente! Un'opzione di cui avrai bisogno, "--tx-out".

11 Le sotto-opzioni di "tx-out" riguardanti lo script di Plutus possono essere saltate. Non ne hai bisogno per ora.

12 Stai iniziando gradualmente a comprendere. Nessun multiasset, nessun NFT, nessuno script di Plutus per le tue transazioni.

```
[
  --read-only-tx-in-reference <TX-IN>
  | --tx-in-collateral <TX-IN>
  | --tx-out-return-collateral <ADDRESS VALUE>
  | --tx-total-collateral <INTEGER>
  | --required-signer <FILE> | --required-signer-hash <HASH>
  | --tx-out <ADDRESS VALUE>
]
```

```
[
  --tx-out <ADDRESS VALUE>
  | --tx-out-datum-hash <HASH>
  | --tx-out-datum-hash-cbor-file <CBOR FILE>
  | --tx-out-datum-hash-file <JSON FILE>
  | --tx-out-datum-hash-value <JSON VALUE>
  | --tx-out-datum-embed-cbor-file <CBOR FILE>
  | --tx-out-datum-embed-file <JSON FILE>
  | --tx-out-datum-embed-value <JSON VALUE>
  | --tx-out-inline-datum-cbor-file <CBOR FILE>
  | --tx-out-inline-datum-file <JSON FILE>
  | --tx-out-inline-datum-value <JSON VALUE>
  | --tx-out-reference-script-file <FILE>
]
[--mint <VALUE>]
```

```
[
  --mint <VALUE>
  | --mint-script-file <FILE>
  [
    (
      --mint-redeemer-cbor-file <CBOR FILE>
      | --mint-redeemer-file <JSON FILE>
      | --mint-redeemer-value <JSON VALUE>
    )
    --mint-execution-units <INT, INT>
    --simple-minting-script-tx-in-reference <TX-IN>
    --policy-id <HASH>
  ]
  | --mint-tx-in-reference <TX-IN>
  | --mint-plutus-script-v2
  (
    --mint-reference-tx-in-redeemer-cbor-file <CBOR FILE>
    | --mint-reference-tx-in-redeemer-file <JSON FILE>
    | --mint-reference-tx-in-redeemer-value <JSON VALUE>
  )
  --mint-reference-tx-in-execution-units <INT, INT>
  --policy-id <HASH>
]
```

13 Dovrai usare 3 delle prossime 4 opzioni

14 Ottieni il TTL, la commissione e il file del certificato.

15 Ancora una volta, non saranno utilizzate opzioni relative ai certificati degli script Plutus.

- "--invalid-before" determina da quale Slot la transazione sarà valida per essere elaborata.
- mentre "--invalid-after" determina da quale Slot la transazione diventerà non valida. (proprio come una data di scadenza)

```
[
  --invalid-before <SLOT>
  | --invalid-hereafter <SLOT>
  | --fee <LOVELACE>
  | --certificate-file <CERTIFICATEFILE>
]
```

È per questo motivo che hai eseguito il comando "cardano-cli query tip" in alcuni esercizi precedenti. Conoscendo il numero dello slot del tuo nodo sincronizzato, puoi determinare un "time to live" o "TTL" per la tua transazione mentre si trova nella memoria pool. Quindi ora puoi aggiungere alla tua bozza queste tre opzioni che saranno dettagliate in seguito.

```
[
  --invalid-before <SLOT>
  | --invalid-hereafter <SLOT>
  | --fee <LOVELACE>
  | --certificate-file <CERTIFICATEFILE>
]
```

```
[
  --certificate-file <CERTIFICATEFILE>
  | --certificate-script-file <FILE>
  [
    (
      --certificate-redeemer-cbor-file <CBOR FILE>
      | --certificate-redeemer-file <JSON FILE>
      | --certificate-redeemer-value <JSON VALUE>
    )
    --certificate-execution-units <INT, INT>
    --certificate-tx-in-reference <TX-IN>
    --certificate-plutus-script-v2
  ]
  | --certificate-reference-tx-in-redeemer-cbor-file <CBOR FILE>
  | --certificate-reference-tx-in-redeemer-file <JSON FILE>
  | --certificate-reference-tx-in-redeemer-value <JSON VALUE>
  | --certificate-reference-tx-in-execution-units <INT, INT>
]
```

16 L'opzione "--withdrawal" è un input che ti consente di prelevare le tue ricompense dal tuo stake.addr.

```
--withdrawal <WITHDRAWAL>
[ --withdrawal-script-file <FILE>
  [
    ( --withdrawal-redeemer-cbor-file <CBOR FILE>
    | --withdrawal-redeemer-file <JSON FILE>
    | --withdrawal-redeemer-value <JSON VALUE>
    )
  ]
--withdrawal-execution-units <INT, INT>]]
| --withdrawal-tx-in-reference <TX-IN>
--withdrawal-plutus-script-v2
( --withdrawal-reference-tx-in-redeemer-cbor-file <CBOR FILE>
| --withdrawal-reference-tx-in-redeemer-file <JSON FILE>
| --withdrawal-reference-tx-in-redeemer-value <JSON VALUE>
)
--withdrawal-reference-tx-in-execution-units <INT, INT>
]]
[--json-metadata-no-schema | --json-metadata-detailed-schema]
[--auxiliary-script-file <FILE>]
[--metadata-json-file <FILE> | --metadata-cbor-file <FILE>]
[--genesis <FILE> | --protocol-params-file <FILE>]
[--update-proposal-file <FILE>]
--out-file <FILE>
```

17 Puoi saltare l'opzione "--withdrawal" e quelle correlate allo script di Plutus per ora.

```
[-withdrawal <WITHDRAWAL>
--withdrawal-script-file <FILE>
  [
    ( --withdrawal-redeemer-cbor-file <CBOR FILE>
    | --withdrawal-redeemer-file <JSON FILE>
    | --withdrawal-redeemer-value <JSON VALUE>
    )
  ]
--withdrawal-execution-units <INT, INT>]]
| --withdrawal-tx-in-reference <TX-IN>
--withdrawal-plutus-script-v2
( --withdrawal-reference-tx-in-redeemer-cbor-file <CBOR FILE>
| --withdrawal-reference-tx-in-redeemer-file <JSON FILE>
| --withdrawal-reference-tx-in-redeemer-value <JSON VALUE>
)
--withdrawal-reference-tx-in-execution-units <INT, INT>
]]
[--json-metadata-no-schema | --json-metadata-detailed-schema]
[--auxiliary-script-file <FILE>]
[--metadata-json-file <FILE> | --metadata-cbor-file <FILE>]
[--genesis <FILE> | --protocol-params-file <FILE>]
[--update-proposal-file <FILE>]
--out-file <FILE>
```

18 Restano solo poche opzioni.

Al momento non hai metadati da inviare, né un file di script ausiliario, né la necessità di specificare un file di genesis o parametri di protocollo. Inoltre, non stai inviando una proposta di aggiornamento per il fondo Catalyst. Quello che ti rimane è semplicemente la possibilità di dare un nome al file per la tua bozza di transazione (--out-file <FILE>).

```
[-json-metadata-no-schema | --json-metadata-detailed-schema]
--auxiliary-script-file <FILE>
--metadata-json-file <FILE> | --metadata-cbor-file <FILE>
--genesis <FILE> | --protocol-params-file <FILE>
--update-proposal-file <FILE>]
--out-file <FILE>
```

19 Raggruppando le opzioni che hai copiato durante questo esercizio, otterrai qualcosa del genere:

cardano-cli transaction build-raw

```
--tx-in <TX-IN>
--tx-out <ADDRESS VALUE>
[--invalid-hereafter <SLOT>]
[--fee <LOVELACE>]
[--certificate-file <CERTIFICATEFILE>]
--out-file <FILE>
```

20 Ora per completare la tua bozza:

Aggiungi l'input UTXO (tx-in), l'indirizzo per il resto (tx-out) e il file del certificato. Per ora, assegna il valore 0 a tx-out, invalid-hereafter e fee.

cardano-cli transaction build-raw

```
--tx-in 1234a4d18e9dkhb34234kjbvdec3ad81e299c#0
--tx-out $(cat paymentwithstake.addr)+0
--invalid-hereafter 0
--fee 0
--certificate-file stake.cert
--out-file tx.raw
```

21 Ecco come apparirà sul tuo terminale:

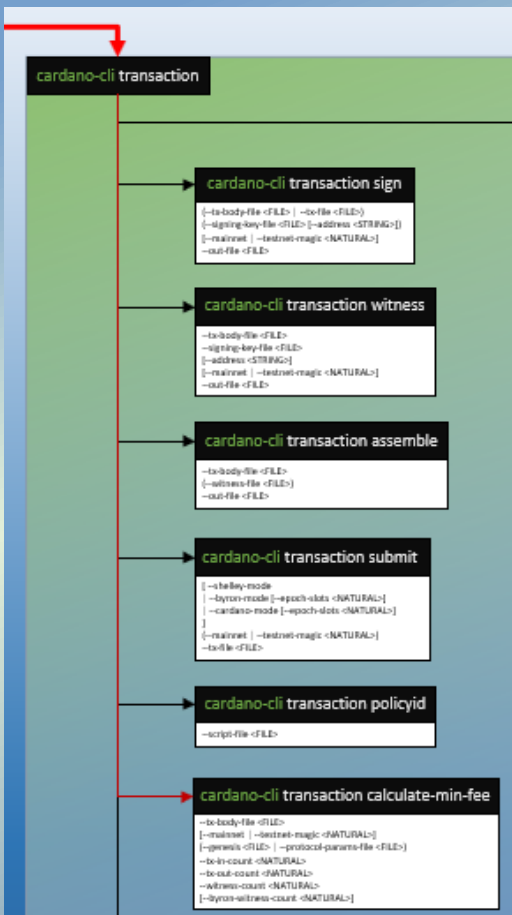
```
user@computer:~$ cardano-cli transaction build-raw \
> --tx-in 1234a4d18e9dkhb34234kjbvdec3ad81e299c#0 \
> --tx-out $(cat paymentwithstake.addr)+0 \
> --invalid-hereafter 0 \
> --fee 0 \
> --certificate-file stake.cert \
> --out-file tx.raw
```

Congratulazioni! Ce l'hai fatta. Salva il comando e le opzioni dal punto 21 in un editor di file di testo, ne avrai bisogno dopo il prossimo esercizio. Ora calcolerai le commissioni (fees) che la tua transazione ti costerà. Quindi potrai sottrarle dall'importo del tuo UTXO (tx-in) e non dimenticare di includere il deposito per la registrazione dell'indirizzo di stake.

Nono esercizio: Calcolo delle commissioni (fees)

Air Gap

1 Trova il comando che utilizzerai per il calcolo delle commissioni (fees).



2 Hai 9 opzioni in totale.

Questo comando ti fornirà esattamente l'importo delle commissioni (fees) che dovrai pagare in base al numero di tx-in, tx-out e al numero di firme richieste.

cardano-cli transaction calculate-min-fee

```
--tx-body-file <FILE>
[--mainnet | --testnet-magic <NATURAL>]
(--genesis <FILE> | --protocol-params-file <FILE>)
--tx-in-count <NATURAL>
--tx-out-count <NATURAL>
--witness-count <NATURAL>
[--byron-witness-count <NATURAL>]
```

3 Solo 3 di queste opzioni non saranno utilizzate.

- testnet-magic (ovviamente useremo la mainnet in questo tutorial)
- genesis (utilizzerai i parametri di protocollo che hai ottenuto in precedenza)
- byron-witness-count (perché non utilizzi coppie di chiavi dell'era byron)

cardano-cli transaction calculate-min-fee

```
--tx-body-file <FILE>
[--mainnet | --testnet-magic <NATURAL>]
(--genesis <FILE> | --protocol-params-file <FILE>)
--tx-in-count <NATURAL>
--tx-out-count <NATURAL>
--witness-count <NATURAL>
[-byron-witness-count <NATURAL>]
```

4 Solo 3 di queste opzioni saranno utilizzate.

Specificare il numero di indirizzi di input e output, nonché il numero di chiavi che utilizzerai per firmare la tua transazione.

cardano-cli transaction calculate-min-fee

```
--tx-body-file <FILE>
[--mainnet | --testnet-magic <NATURAL>]
(--genesis <FILE> | --protocol-params-file <FILE>)
--tx-in-count 1
--tx-out-count 1
--witness-count 2
[-byron-witness-count <NATURAL>]
```

5 Quindi devi semplicemente indicare il PERCORSO al tuo file protocol.json e alla tua bozza di transazione tx.raw.

cardano-cli transaction calculate-min-fee

```
--tx-body-file tx.raw
--mainnet
--protocol-params-file protocol.json
--tx-in-count 1
--tx-out-count 1
--witness-count 2
```

6 Questo è il risultato nel tuo terminale. (L'importo delle commissioni - fees - non è sempre lo stesso.)

```
user@computer:~$ cardano-cli transaction calculate-min-fee \
> --tx-body-file tx.raw \
> --mainnet \
> --protocol-params-file protocol.json \
> --tx-in-count 1 \
> --tx-out-count 1 \
> --witness-count 2
```

Se il comando funziona come previsto, le commissioni verranno visualizzate in fondo ad esso.

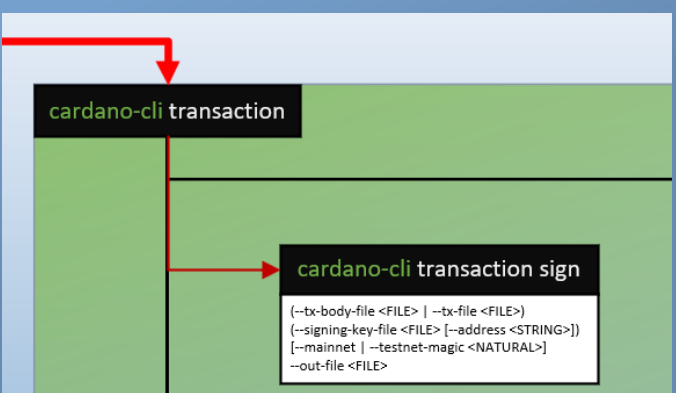
```
user@computer:~$ cardano-cli transaction calculate-min-fee \
> --tx-body-file tx.raw \
> --mainnet \
> --protocol-params-file protocol.json \
> --tx-in-count 1 \
> --tx-out-count 1 \
> --witness-count 2
178525 Lovelace
```

Per il prossimo esercizio dovrai aprire il file con il tuo editor di testo che hai salvato in precedenza con il comando "cardano-cli transaction build-raw", nell'esercizio otto. Modificherai il suo contenuto per creare la tua transazione finale.

Decimo esercizio: Costruzione della transazione finale Air Gap

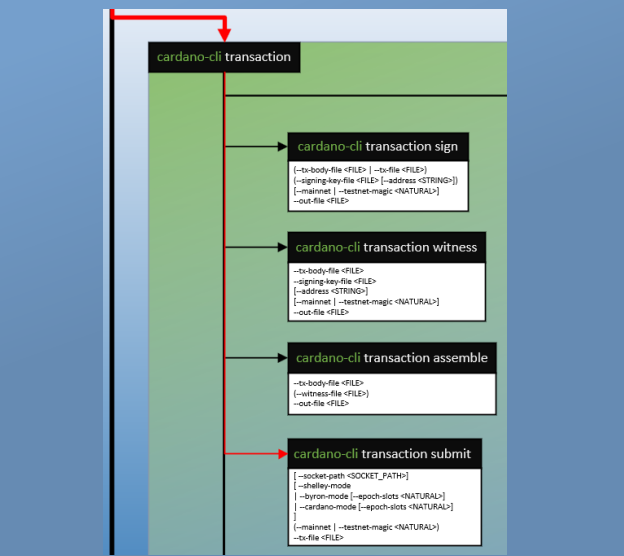
<p>1 Questo è la tua bozza della transazione dall'esercizio otto.</p> <p>La modificherai per inserire l'importo delle commissioni - fees - (che conosci) e quindi calcolerai l'importo di Lovelace da inviare al tuo indirizzo.</p> <pre>cardano-cli transaction build-raw --tx-in 1234a4d18e9dkhb34234kjbvdec3ad81e299c#0 --tx-out \$(cat paymentwithstake.addr)+0 --invalid-hereafter 0 --fee 178525 --certificate-file stake.cert --out-file tx.raw</pre>	<p>2 Utilizzando il comando "expr" puoi eseguire il calcolo.</p> <p>Ammontare dell'UTXO Deposito indirizzo di stake</p> <pre>user@computer:~\$ expr 10000000 - 178525 - 2000000</pre> <p style="text-align: center;">↑ commissione</p> <pre>user@computer:~\$ expr 10000000 - 178525 - 2000000 7821475 user@computer:~\$</pre>	<p>3 Puoi inserire il risultato nella tua transazione.</p> <p>Nota che non deve esserci spazio tra il tuo indirizzo, l'operatore "+" e l'importo in Lovelace. Altrimenti, si verificherà un errore durante l'esecuzione del comando.</p> <pre>cardano-cli transaction build-raw --tx-in 1234a4d18e9dkhb34234kjbvdec3ad81e299c#0 --tx-out \$(cat paymentwithstake.addr)+7821475 --invalid-hereafter 0 --fee 178525 --certificate-file stake.cert --out-file tx.raw</pre>
<p>4 Ora determiniamo il tuo "TTL" (time-to-live)</p> <p>Per scegliere da quale Slot la transazione diventerà invalida, devi conoscere il numero di Slot in cui ti trovi, che puoi ottenere ripetendo l'esercizio #6 o controllando i tuoi logs. Ecco un esempio di quello che potresti ottenere:</p> <pre>{ "block": 8749178, "epoch": 410, "era": "Babbage", "hash": "367e4af96abc18e1d4b5de08af535cb508e691...", "slot": 92029934, "syncProgress": "100.00" }</pre>	<p>5 Aggiungi alcuni minuti ad esso. (1 slot = 1 secondo)</p> <p>Per permetterti di avere il tempo di firmare la tua transazione e inviarla sul tuo "hot node", aggiungi 15 minuti al valore dell'opzione. (92029934 + 900 = 92030834)</p> <pre>cardano-cli transaction build-raw --tx-in 1234a4d18e9dkhb34234kjbvdec3ad81e299c#0 --tx-out \$(cat paymentwithstake.addr)+7821475 --invalid-hereafter 92030834 --fee 178525 --certificate-file stake.cert --out-file tx.raw</pre>	<p>6 Questo è il risultato nel tuo terminale:</p> <pre>user@computer:~\$ cardano-cli transaction build-raw \ > --tx-in 1234a4d18e9dkhb34234kjbvdec3ad81e299c#0 \ > --tx-out \$(cat paymentwithstake.addr)+7821475 \ > --invalid-hereafter 92030834 \ > --fee 178525 \ > --certificate-file stake.cert \ > --out-file tx.raw</pre>

Undicesimo esercizio: Firmare la tua transazione Air Gap

<p>1 Ora sei pronto per firmare la tua transazione con le tue 2 chiavi private (payment.skey e stake.skey).</p>  <pre>cardano-cli transaction sign [--tx-body-file <FILE> --tx-file <FILE>] [--signing-key-file <FILE> [-address <STRING>]] [--mainnet --testnet-magic <NATURAL>] --out-file <FILE></pre>	<p>2 Hai un totale di 7 opzioni.</p> <p>Per questo, dovrai menzionare il percorso al tuo "file di transazione", i percorsi delle tue 2 chiavi private, la rete da utilizzare e il nome del file che invierai alla blockchain.</p> <pre>cardano-cli transaction sign --tx-body-file <FILE> --tx-file <FILE> --signing-key-file <FILE> [-address <STRING>] --mainnet --testnet-magic <NATURAL> --out-file <FILE></pre>	<p>3 Questo è il risultato sul tuo terminale:</p> <pre>user@computer:~\$ cardano-cli transaction sign \ > --tx-body-file tx.raw \ > --signing-key-file payment.skey \ > --signing-key-file stake.skey \ > --mainnet \ > --out-file tx.signed</pre> <div style="border: 1px solid red; padding: 5px; margin-top: 10px;"> <p>Nota che in molte situazioni è possibile utilizzare alcune opzioni più di una volta.</p> </div>
--	---	---

Puoi ora trasferire il file "tx.signed" al tuo "Hot Node" per inviarlo alla blockchain, ma prima assicurati che gli accessi a questo file siano impostate come "sola lettura".

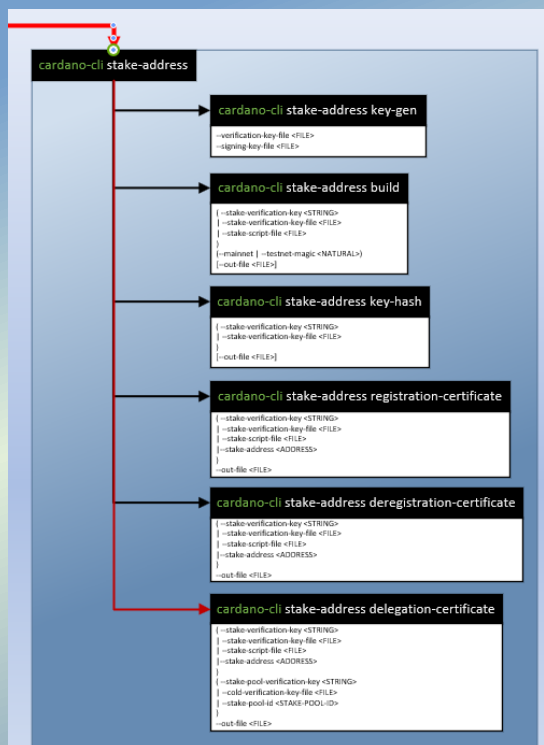
Dodicesimo esercizio: Invia la tua transazione Hot Node

<p>1 Ora sei pronto ad inviare la tua transazione!</p>  <pre>cardano-cli transaction submit [--socket-path <SOCKET_PATH>] [--shelley-mode] [--byron-mode [-epoch-slots <NATURAL>]] [--cardano-mode [-epoch-slots <NATURAL>]]] --mainnet --testnet-magic <NATURAL> --tx-file <FILE></pre>	<p>2 Hai un totale di 9 opzioni.</p> <p>L'opzione "--socket-path" non è necessaria se il percorso al file socket del tuo nodo è già presente nell'ambiente. Userai solo ciò che è richiesto. In altre parole, la rete e il nome del file da inviare.</p> <pre>cardano-cli transaction submit [--socket-path <SOCKET_PATH>] [--shelley-mode] [--byron-mode [-epoch-slots <NATURAL>]] [--cardano-mode [-epoch-slots <NATURAL>]]] --mainnet --testnet-magic <NATURAL> --tx-file <FILE></pre>	<p>3 Questo è il comando nel terminale:</p> <pre>user@computer:~\$ cardano-cli transaction submit \ > --mainnet \ > --tx-file tx.signed</pre> <p style="text-align: center;">↓</p> <pre>user@computer:~\$ cardano-cli transaction submit \ > --mainnet \ > --tx-file tx.signed transaction successfully submitted</pre>
--	---	--

Congratulazioni, il tuo indirizzo di stake è ora registrato sulla blockchain. Ora puoi creare un certificato di delega per scegliere una stake pool e partecipare al protocollo "Proof of Stake" di Cardano. Tuttavia, prima di passare al prossimo esercizio, assicurati di eliminare il file tx.signed dal tuo nodo attivo. (Non ne avrai più bisogno)

Tredicesimo esercizio: Creazione di un certificato di delega Air Gap

1 Prima di tutto, individua il comando che utilizzerai per il tuo certificato.



2 Hai 8 opzioni in totale.

Una volta inviato alla blockchain, questo certificato verrà utilizzato per indicare con quale pool vorrai mettere in stake i tuoi ADA. Ci sono 2 gruppi di opzioni necessarie. Queste opzioni saranno scelte in modo diverso a seconda delle tue esigenze (ad esempio, utilizzando cold.vrf per il tuo stake pool personale). In questo esercizio, supponiamo che tu voglia scegliere tra diverse opzioni di stake pool.

```
cardano-cli stake-address delegation-certificate
[ --stake-verification-key <STRING>
  | --stake-verification-key-file <FILE>
  | --stake-script-file <FILE>
  | --stake-address <ADDRESS>
  |
  | --stake-pool-verification-key <STRING>
  | --cold-verification-key-file <FILE>
  | --stake-pool-id <STAKE-POOL-ID>
  |
  | --out-file <FILE> ]
```

3 Usiamo quindi --stake-address e --stake-pool-id

- Specifica il percorso (PATH) del tuo stake.addr.
- L'ID della stake pool a cui desideri delegare il tuo portafoglio. (può essere codificato in Bech32 o esadecimale)
- Il nome del tuo file del certificato.

```
cardano-cli stake-address delegation-certificate
[ --stake-verification-key <STRING>
  | --stake-verification-key-file <FILE>
  | --stake-script-file <FILE>
  | --stake-address stake.addr
  |
  | --stake-pool-verification-key <STRING>
  | --cold-verification-key-file <FILE>
  | --stake-pool-id pool1mt8sdg37f2h3rypyuc77k7vxrjshtvjw04zdljae9vdzyt9uu34
  |
  | --out-file delegation.cert ]
```

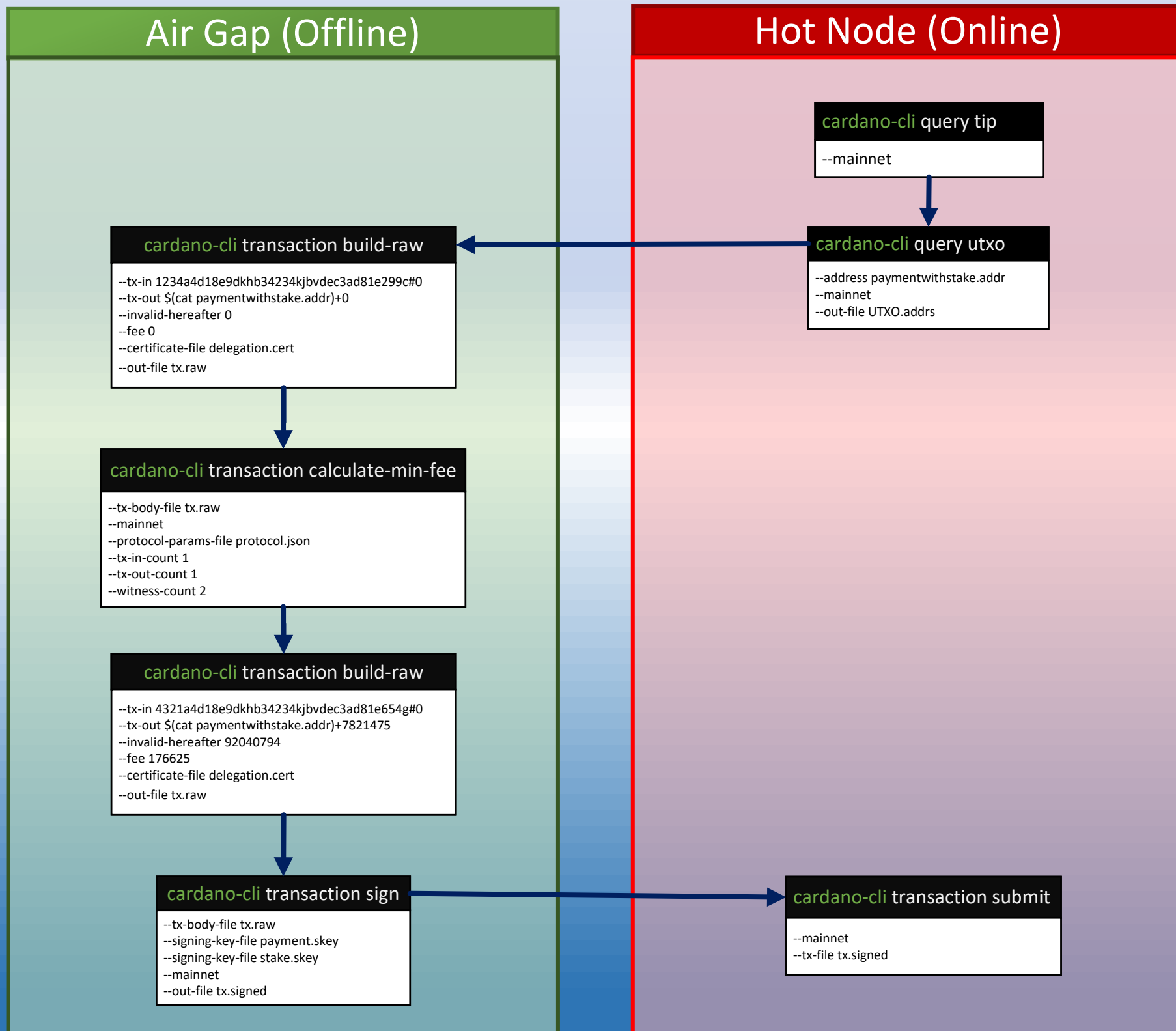
4 Questo è il risultato sul tuo terminale:

```
user@computer:~$ cardano-cli stake-address delegation-certificate \
> --stake-address stake.addr \
> --stake-pool-id pool1mt8sdg37f2h3rypyuc77k7vxrjshtvjw04zdljae9vdzyt9uu34 \
> --out-file delegation.cert
```

! Puoi ottenere l'ID della stake pool tramite cexplorer.io oppure, se ti piace questo documento, faccelo sapere e potremmo aggiungere i comandi "query stakepools", "query pool-state" e "query pool-distribution" al secondo capitolo di questo tutorial.

Puoi ora ripetere gli esercizi da 6 a 12, assicurandoti di sostituire il file stake.cert con il file delegation.cert quando costruisci la tua transazione. E non dimenticare che quando calcoli le commissioni non devi considerare il deposito per l'indirizzo di stake di 2 ADA (che è già stato fatto in precedenza).

Riepilogo delle operazioni: Processo di invio del certificato di delega



Concluderemo la parte 1 di questo tutorial con una citazione da un collega SPO che apprezzo molto:
"Dovremmo incoraggiare i nuovi SPO, anche se hanno basse competenze. Impareranno, e Cardano si decentralizzerà."
 -- @StakeWithPride