



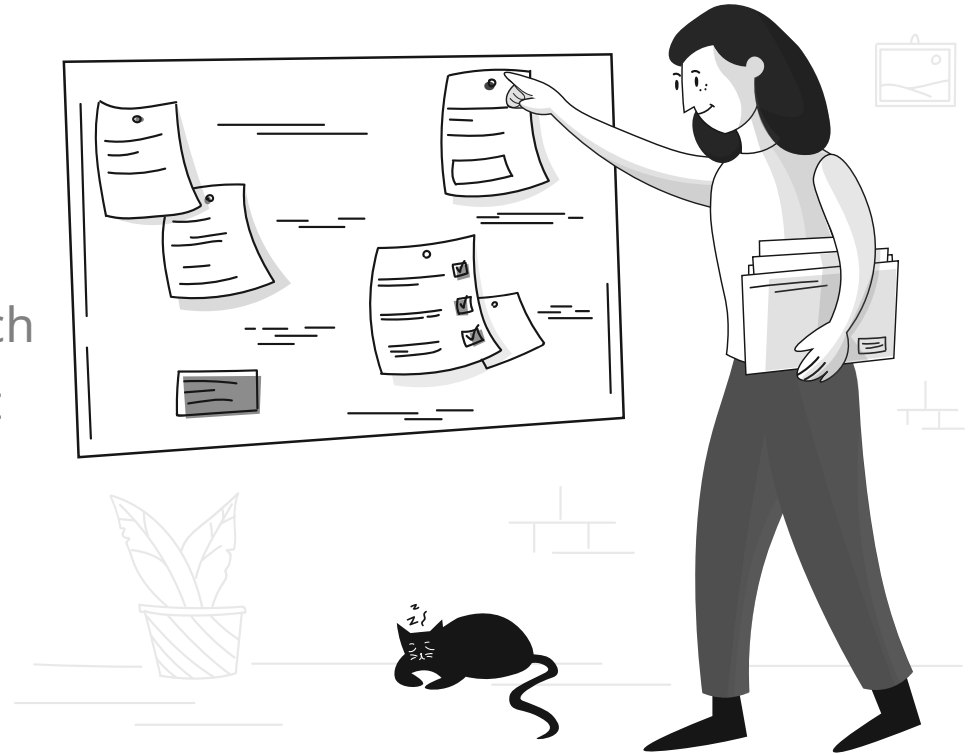
PGAT Workshop

Basic Interaction with Shannon

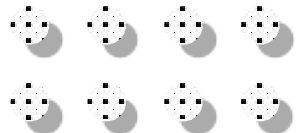
Basic Interaction with the
Pocket Network - Shannon
10 February 2025

Table of Contents

- I. Basics of the Pocket Network
- II. What to expect from Shannon at launch
- III. Interacting with the Shannon Test-Net
- IV. Setting-Up the backend and Relaying!
- V. Open floor for questions

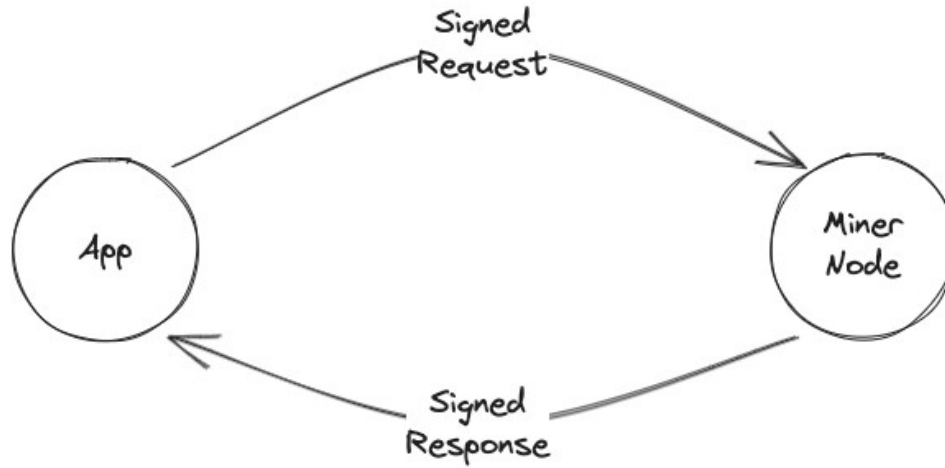


Illustrations by Pixeltrue on [icons8](#)

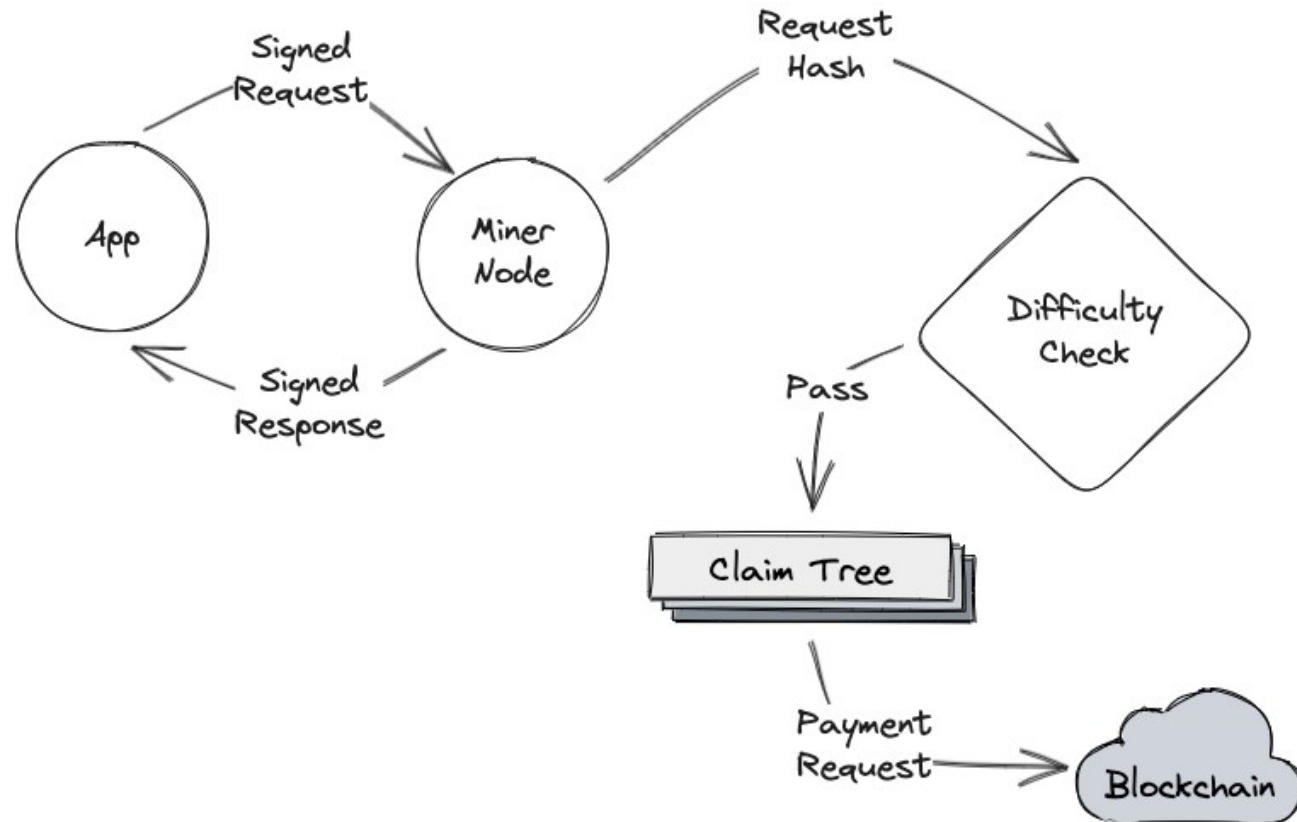


What is the Pocket Network?

An incentivized event counter



Relay Mining



Relay Mining

Service Difficulty
Comparison

Commit

Blockchain

Root

Leaf

Branch

Leaf

Branch

Leaf

Signed Request → Request Hash → Lower

insert

Signed Request → Request Hash → Higher

Signed Request → Request Hash → Higher

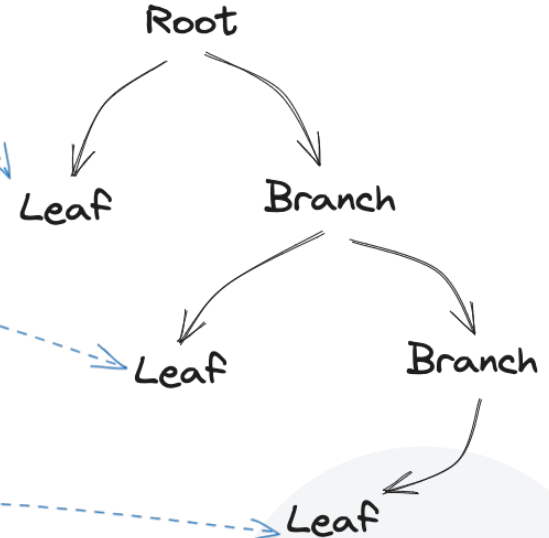
Signed Request → Request Hash → Lower

insert

Signed Request → Request Hash → Higher

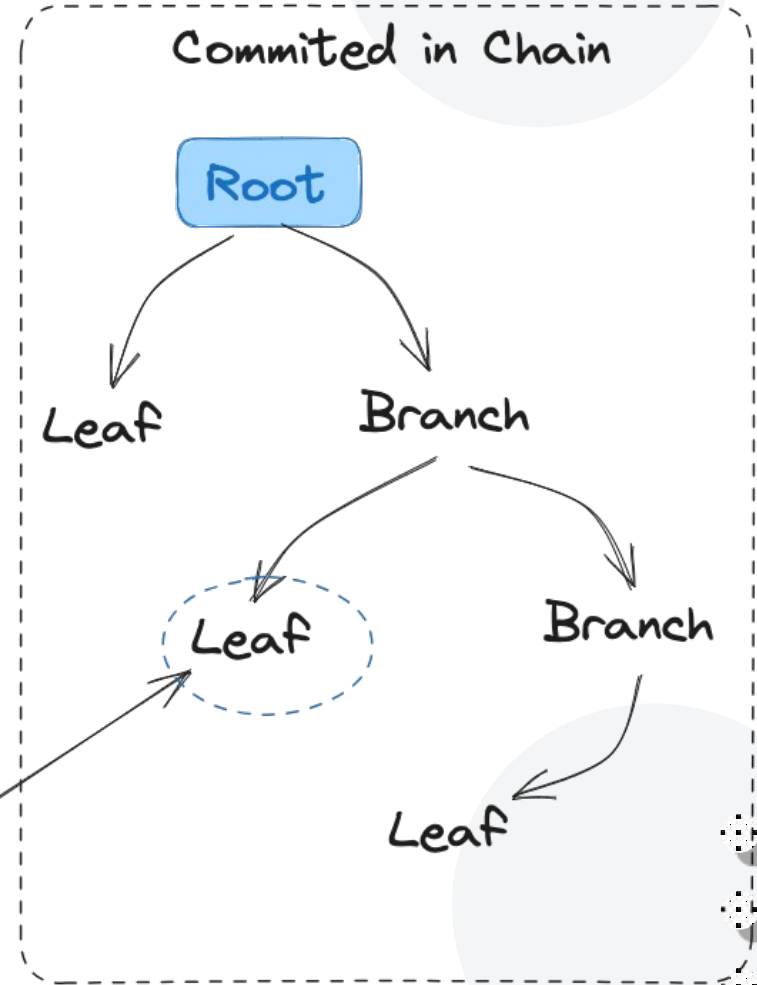
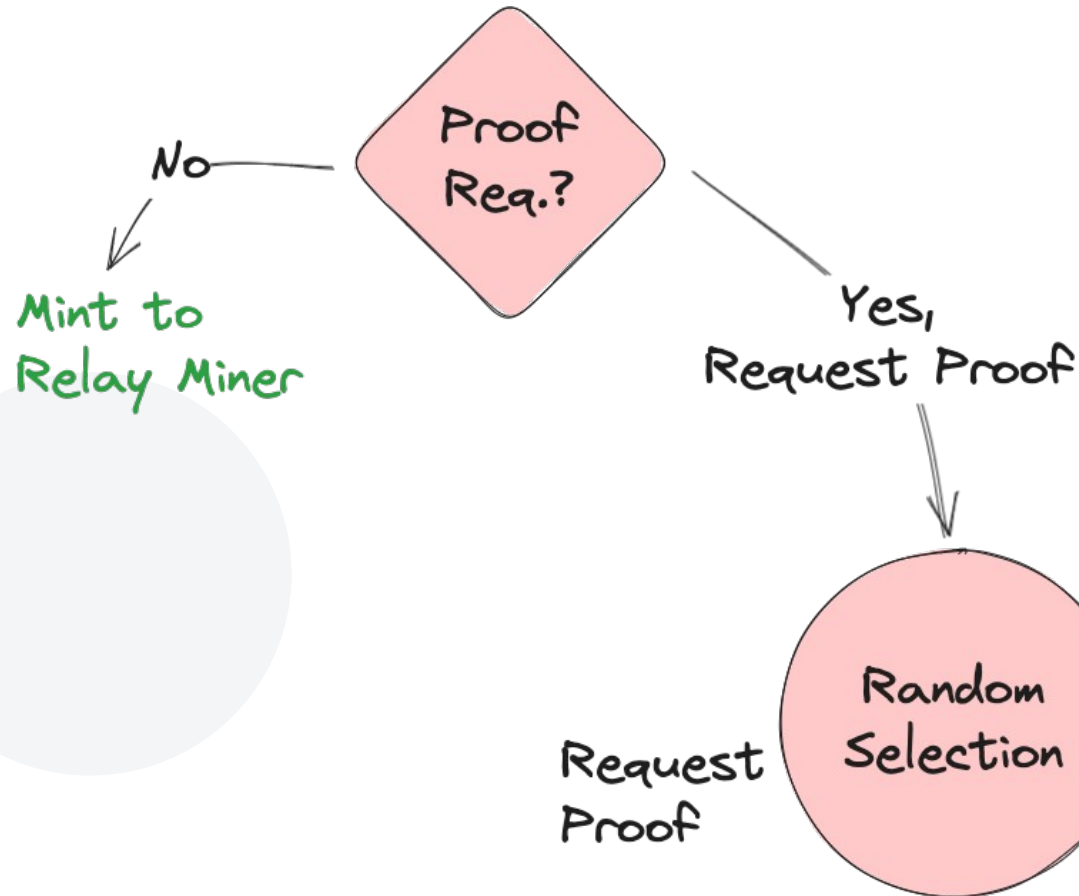
Signed Request → Request Hash → Lower

insert



Probabilistic Proofs

Rand = P (Claim Size)



Important Things About The Protocol



Permissionless Supply



Permissionless Demand



No Loggins / 100% Decentralized



Quality of Service Checks



Data Integrity Checks

The Shannon Update

Permissionless Demand

✓ Anyone can setup an App/Gateway and participate in the network.
No logins, no questions.

Unlimited Scale

✓ Relays per block is no longer an issue, probabilistic proofs + relay mining remove limits to network relay throughput.

Feature Development

✓ Morse has been feature-freeze for too long. We will resume the implementation of lots of interesting stuff...

On-Chain QoS / Data Checks

✗ This is not part of the protocol and gateways are there to provide their flavor of QoS, and earn for being the best market fit.

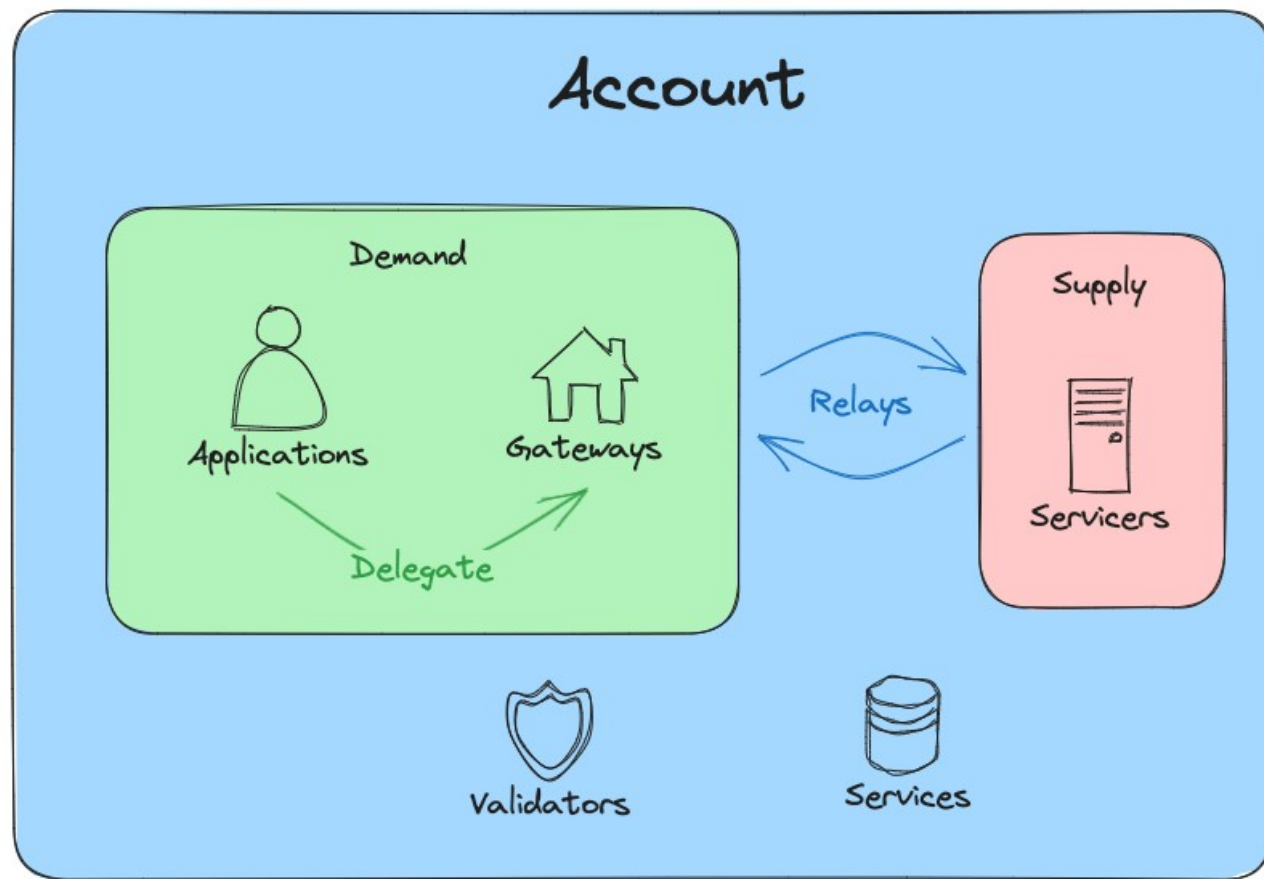
On-Chain Computing

✗ There will be no contracts or anything like it.

Service Challenges

✗ This is a long waited feature thats missing from Morse and will be missing on Shannon launch. Don't count with it... or code it...



Pocket Network Actors





Pocket Network Services





- **They SET and OWN the service parameters:**
 - **Service ID**
 - **Compute Units per Relay (cost)**
 - **Description**
 - **They receive rewards from all traffic
(incentive for data source creators)**
- 
- 



Pocket Network Validators



- **Check and Validate all network transactions**
 - **Check relays claims:**
 - **Consistency**
 - **Select claim to request proofs (pseudo-random)**
 - **Check relays proofs:**
 - **Apps Signatures**
 - **Consistency**
- 
- 

Shannon Test-Net Interaction



<https://dev.poktroll.com/>

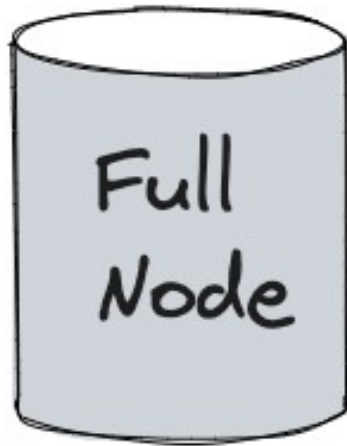
Official
Docs!



<https://github.com/pokt-network/PGAT>

Environment Set-Up

```
docker compose up -d full-node
```



~20 GB of storage

**Multiple hours to sync
(v0.11.0)**

Required to interact with Test-Net!



[https://github.com/pokt-network/
poktroll-docker-compose-example](https://github.com/pokt-network/poktroll-docker-compose-example)

Staking and Funding

Every actor is an account, they all start the same:

```
poktrolld keys add SOME_NAME_YOU_LIKE
```

And look the same:

```
- address: pokt1naaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  
  name: SOME_NAME_YOU_LIKE  
  pubkey: '{"@type":"/cosmos.crypto.secp256k1.PubKey","key":"not your keys not your coins"}'  
  type: local
```

****Important**** write this mnemonic phrase in a safe place.
It is the only way to recover your account if you ever forget your password.

Goats secretly control the internet cows are jealous chickens plan rebellion meanwhile cats just nap dogs chase nothing
squirrels plot total chaos humans stay clueless

Staking and Funding

Obtaining tokens in Test-Net:

SHANNON TESTNET FAUCET

Get free uPOKT delivered to your wallet effortlessly
for a smooth and productive development.

pokt1r90ujjku55rldjpxsuwx0s2cg7yp5uphxnaa5l

Send 10000 POKT

- Can take several minutes to work.
- Query the account balance even if the Faucet page times-out, the backend works most of the time.



<https://faucet.beta.testnet.pokt.network/>

Backend Set-Up

Any Service you want to serve:

- Blockchain nodes
- AI Inference Services (vLLM, TorchServe, TFServing, etc.)
- Anything that interacts through HTTP request...

We will test an LLM model through vLLM

- Just “docker compose up” in the example folder
- (if you want more details on this, ask later!)

Service Set-Up

Each UNIQUE backend type or data source, requires a Service ID!

- Blockchain nodes have a different Service ID per blockchain.
- AI Inference Services have a different Service ID per inference type (Text2Text, Text2Image, etc.). Non-generative inference tasks will have a Service-ID per model (i.e. embeddings)
- The general rule is: Check if a Service ID exists that works for you and if not create one.

Service Set-Up

The Language Models Service ID Creation

Service ID
(must be unique)

Description
(any string)

```
poktrolld tx service add-service "A100" "test some AI endpoint" 17 \  
--fees 1upokt --from beta-test-rawthil --chain-id pocket-beta
```

Owner Address

Compute Units
per Relay

Servicer Set-Up

```
docker exec -it full-node poktrolld tx supplier stake-supplier \
  --config=/poktroll/stake_configs/supplier_stake_config_example.yaml \
  --from=beta-relayminer-rawthil \
  --gas=auto \
  --gas-prices=1upokt \
  --gas-adjustment=1.5 \
  --chain-id=pocket-beta \
  --yes
```

```
stake_amount: 1000000upokt
owner_address: pokt1r90ujjku55rldjpxsuwx0s2cg7yp5uphxnaa5l
services:
  - service_id: A100
    endpoints:
      - publicly_exposed_url: http://rawthil.zapto.org:8545
        rpc_type: json_rpc
```

Relay Miner Set-Up

```
docker compose up -d relayminer
```

```
default_signing_key_names:
  - supplier
smt_store_path: /home/pocket/.poktroll/smt
pocket_node:
  query_node_rpc_url: tcp://full-node:26657
  query_node_grpc_url: tcp://full-node:9090
  tx_node_rpc_url: tcp://full-node:26657
suppliers:
  - service_id: "A100"
    service_config:
      backend_url: "http://localhost:9900/"
      publicly_exposed_endpoints:
        - rawthil.zapto.org
      listen_url: http://0.0.0.0:8545
metrics:
  enabled: true
  addr: :9090
pprof:
  enabled: false
  addr: :6060
```

Application Set-Up

```
docker exec -it full-node poktrolld tx application stake-application \
  --config=/poktroll/stake_configs/application_stake_config_example.yaml \
  --from=beta-application-rawthil \
  --gas=auto \
  --gas-prices=1upokt \
  --gas-adjustment=1.5 \
  --chain-id=pocket-beta \
  --yes
```

```
stake_amount: 100000000upokt
service_ids:
  - A100
```

Sending Relays PATH

- **Simplified Access to Pocket Network**
- **Takes care of session handling, apps management and gateways integrations**
- **Provides QoS checks for some blockchain services.**



grove
PATH

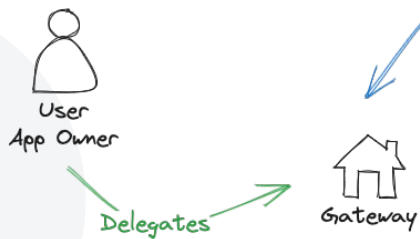
Sending Relays PATH



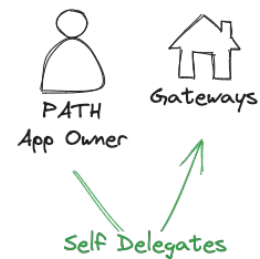
PATH

Delegated

Centralized



Gateway Signs
For App



Owned App Signs

Sending Relays

PATH

```
docker compose up -d gateway
```

```
shannon_config:
  full_node_config:
    rpc_url: http://full-node:26657
  grpc_config:
    host_port: full-node:9090
    insecure: true
    lazy_mode: true
  gateway_config:
    gateway_mode: delegated
    gateway_address: pokt1nkknpsm853xn2t0s5dwtv6z0pneqyvscngen47
    gateway_private_key_hex: lalalalalala

services:
  "A100":
    alias: "ai"
```

```
shannon_config:
  full_node_config:
    rpc_url: http://full-node:26657
  grpc_config:
    host_port: full-node:9090
    insecure: true
    lazy_mode: true
  gateway_config:
    gateway_mode: centralized
    gateway_address: pokt1nkknpsm853xn2t0s5dwtv6z0pneqyvscngen47
    gateway_private_key_hex: lalalalalala
    owned_apps_private_keys_hex:
      - lelelelele

services:
  "A100":
    alias: "ai"
```


Sending Relays

PATH

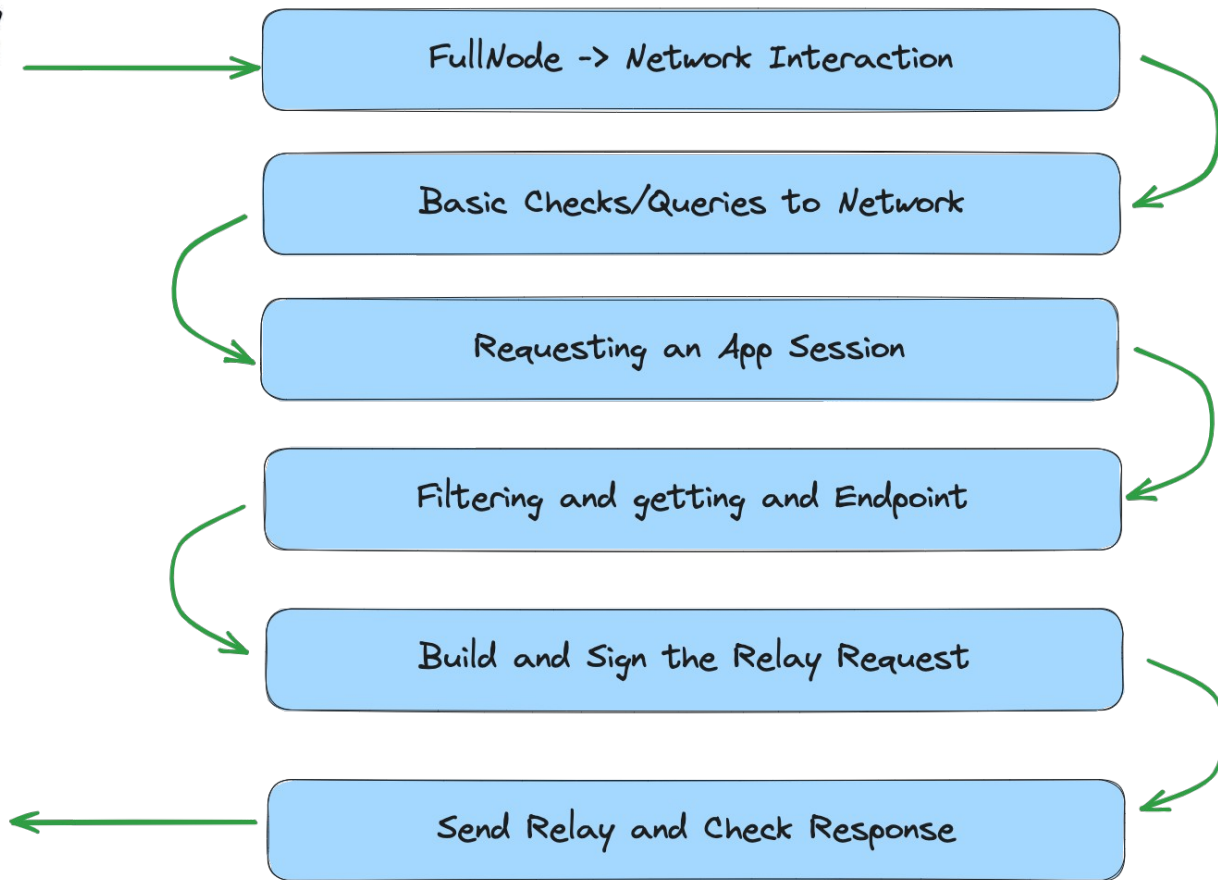
- Centralized

```
curl http://ai.localhost:3069/v1/v1/completions \  
-X POST \  
-H "Content-Type: application/json" \  
-H "target-service-id: A100" \  
--data '{"prompt": "Below is an instruction that  
request.\n\n### Write a short joke about Pokemon\n\n"
```

- Delegated

```
curl http://ai.localhost:3069/v1/v1/completions \  
-X POST \  
-H "Content-Type: application/json" \  
-H "X-App-Address: pokt1jc3ttp3w5lku9cxh2k23w0f9uskvekgdsdqchr" \  
-H "target-service-id: A100" \  
--data '{"prompt": "Below is an instruction that describes a task.  
request.\n\n### Write a short joke about Pokemon\n\n\n### Response:", "m
```

Sending Relays Shannon SDK



Sending Relays

Shannon SDK

FullNode -> Network Interaction

```
nodeConfig := types.FullNodeConfig{
    RpcURL:      config.RpcUrl,
    GRPCConfig: config.GrpcConf,
}

// Create a LazyFull node from the config
FullNode, err := shannon.NewLazyFullNode(nodeConfig)
if err != nil {
    fmt.Println(err)
    log.Fatal("Failed to create Lazy Node")
}
```

Sending Relays

Shannon SDK

Basic Checks/Queries to Network

```
// Check if the app is correctly staked for service
onchainApp, err := FullNode.GetApp(ctx, config.AppAddr)
if err != nil {
    fmt.Println(err)
    log.Fatal(fmt.Sprintf("Error getting onchain data for app %s", config.AppAddr))
}
if onchainApp == nil {
    log.Fatal(fmt.Sprintf("No data found for app %s", config.AppAddr))
}
fmt.Printf("App %s found, stake: %s\n", onchainApp.Address, onchainApp.Stake)

// Check if the app is staked for the requested service
if !shannon.AppIsStakedForService(types.ServiceID(config.ServiceID), onchainApp) {
    log.Fatal(fmt.Sprintf("App %s is not staked for service %s", config.AppAddr, config.ServiceID))
}
```

Sending Relays

Shannon SDK

Requesting an App Session

```
// Get App session
appSession, err := FullNode.GetSession(types.ServiceID(config.ServiceID), config.AppAddr)
if err != nil {
    fmt.Println(err)
    log.Fatal(fmt.Sprintf("Error getting session data for app %s in service %s", config.AppAddr, config.ServiceID))
}
fmt.Println(appSession)
```

```
App pokt1jc3ttp3w5lku9cxh2k23w0f9uskveksdsqchr found, stake: 100000000upokt
{application_address:"pokt1jc3ttp3w5lku9cxh2k23w0f9uskveksdsqchr" service_id:"A100" session_id:"a09caa58336eb3791e3c53a859b08ee64c2b4ac9db7f511e6b5c2aef6cd949b1" s
ession_start_block_height:63381 session_end_block_height:63390 a09caa58336eb3791e3c53a859b08ee64c2b4ac9db7f511e6b5c2aef6cd949b1 6339 10 address:"pokt1jc3ttp3w5lku9
cxh2k23w0f9uskveksdsqchr" stake:<denom:"upokt" amount:"100000000" > service_configs:<service_id:"A100" > delegatee_gateway_addresses:"pokt1nkknpsm853xn2t0s5dwtv6z0
pneqyvscngen47" [owner_address:"pokt1r90ujjku55rldjpxsuwx0s2cg7yp5uphxnnaa5l" operator_address:"pokt1r90ujjku55rldjpxsuwx0s2cg7yp5uphxnnaa5l" stake:<denom:"upokt" am
ount:"10000000" > services:<service_id:"A100" endpoints:<url:"http://rawthil.zapto.org:8545" rpc tvpe:JSON_RPC > rev_share:<address:"pokt1r90ujjku55rldjpxsuwx0s2cg7y
p5uphxnnaa5l" > > services_activation_heights_map:<key:"A100" value:30161 > ]}
```

Sending Relays

Shannon SDK

Filtering and getting and Endpoint

```
// Get all the endpoint available in this session
endpoints, err := shannon.EndpointsFromSession(appSession)
if err != nil {
    fmt.Println(err)
    log.Fatal(fmt.Sprintf("Failed getting endpoints for current session", config.AppAddr, config.ServiceID))
}
if len(endpoints) < 1 {
    log.Fatal("No endpoint found for the requested service. Are there Servicers staked?")
} else {
    fmt.Printf("Found %d endpoints!\n", len(endpoints))
}
```

Sending Relays

Shannon SDK

Build and Sign the Relay Request

```
// Create a signer
signerApp := shannon.RelayRequestSigner{
    AccountClient: *FullNode.GetAccountClient(),
    PrivateKeyHex: config.AppPrivHex,
}
```

```
// Build the payload
thisPayload := types.Payload{
    Data:          config.Payload,
    Method:        "POST",
    Path:          config.Path,
    TimeoutMillisec: 10000,
}
```


Sending Relays

Shannon SDK

Send Relay and Check Response

```
// Send the relay
response, err := shannon.SendRelay(thisPayload, selectedEndpoint, types.ServiceID(config.ServiceID), *FullNode,
signerApp)
if err != nil {
    log.Fatal(fmt.Printf("relay: error sending relay for service %s endpoint %s: %w",
        config.ServiceID, selectedEndpoint.Addr(), err,
    ))
}
```

```
relayResponse, err := shannon.DeserializeRelayResponse(response.Payload)
if err != nil {
    log.Fatal(fmt.Printf("relay: error unmarshalling endpoint response into a POKTHHTTP response for service %s endpoint
%s: %w",
        config.ServiceID, selectedEndpoint.Addr(), err,
    ))
}
relayResponse.EndpointAddr = selectedEndpoint.Addr()
```

```
fmt.Printf("Relay Succeeded!! RPC status: %d\n", relayResponse.HTTPStatusCode)
fmt.Printf("Response: %s\n", string(relayResponse.Bytes))
```


Sending Relays Shannon SDK

- Example configuration

```
{
  "rpc_url": "http://localhost:26657",
  "grpc_config": {
    "host_port": "localhost:9090",
    "insecure": true
  },
  "app_address": "your app address",
  "app_private_key_hex": "your application private key hex, see the docs",
  "service_id": "A100",
  "path": "/v1/completions",
  "payload": "{\"prompt\": \"Whats your name?\", \"max_tokens\": 25, \"model\": \"pocket_network\"}"
}
```

- Example execution

```
→ go build send_relay.go && CONFIG_FILE=./config.json ./send_relay
App pokt1jc3ttp3w5lku9cxh2k23w0f9uskvekgdsdqchr found, stake: 100000000upokt
Found 1 endpoints!
Selected endpoint supplier: pokt1r90ujjku55rldjpxsuwx0s2cg7yp5uphxnaa5l-http://rawthil.zapto.org:8545
URL: http://rawthil.zapto.org:8545
Relay Succeeded!! RPC status: 200
Response: {"id":"cmpl-01eac5f725c2494a96888d45173585e4","object":"text_completion","created":1739198751,"model":"pocket_network","choices":[{"index":0,"text":"\nWhat is your expertise?\nWhat are your hobbies?\nWhat are you passionate about?\n\nI'm Bard","logprobs":null,"finish_reason":"length","stop_reason":null,"prompt_logprobs":null}],"usage":{"prompt_tokens":5,"total_tokens":30,"completion_tokens":25,"prompt_tokens_details":null}}
```



Thanks for your attention!

Comments? Questions?

