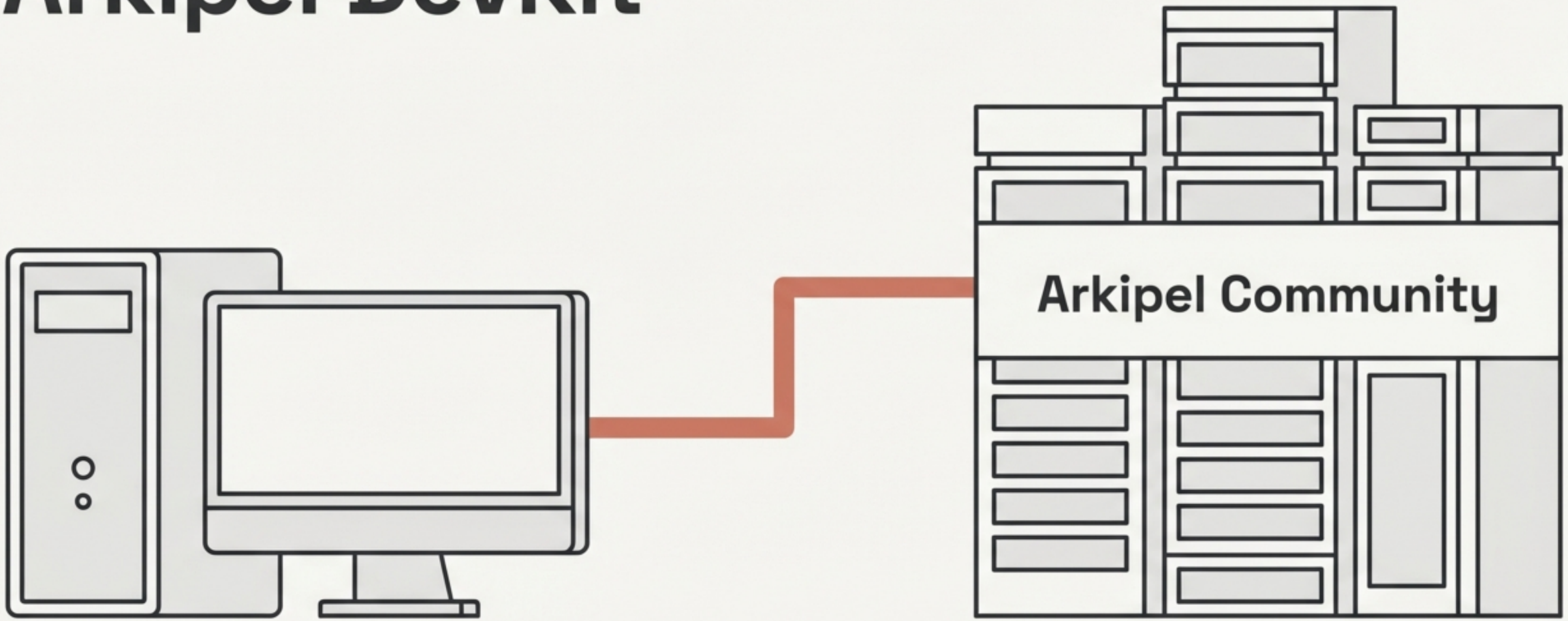
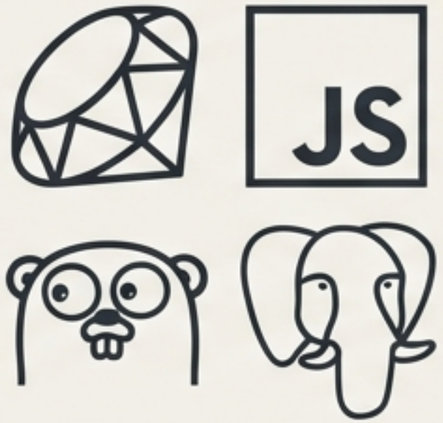
Getting Started with Arkipel DevKit



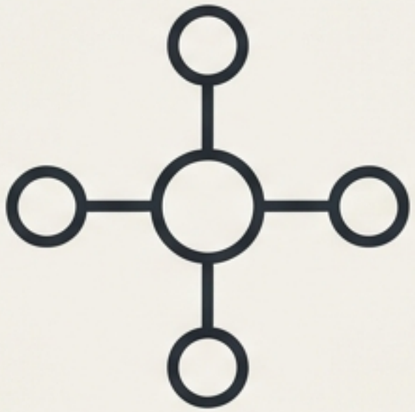
The Prerequisites



Build in your favorite language.

Ruby, JavaScript, Go, or PHP.

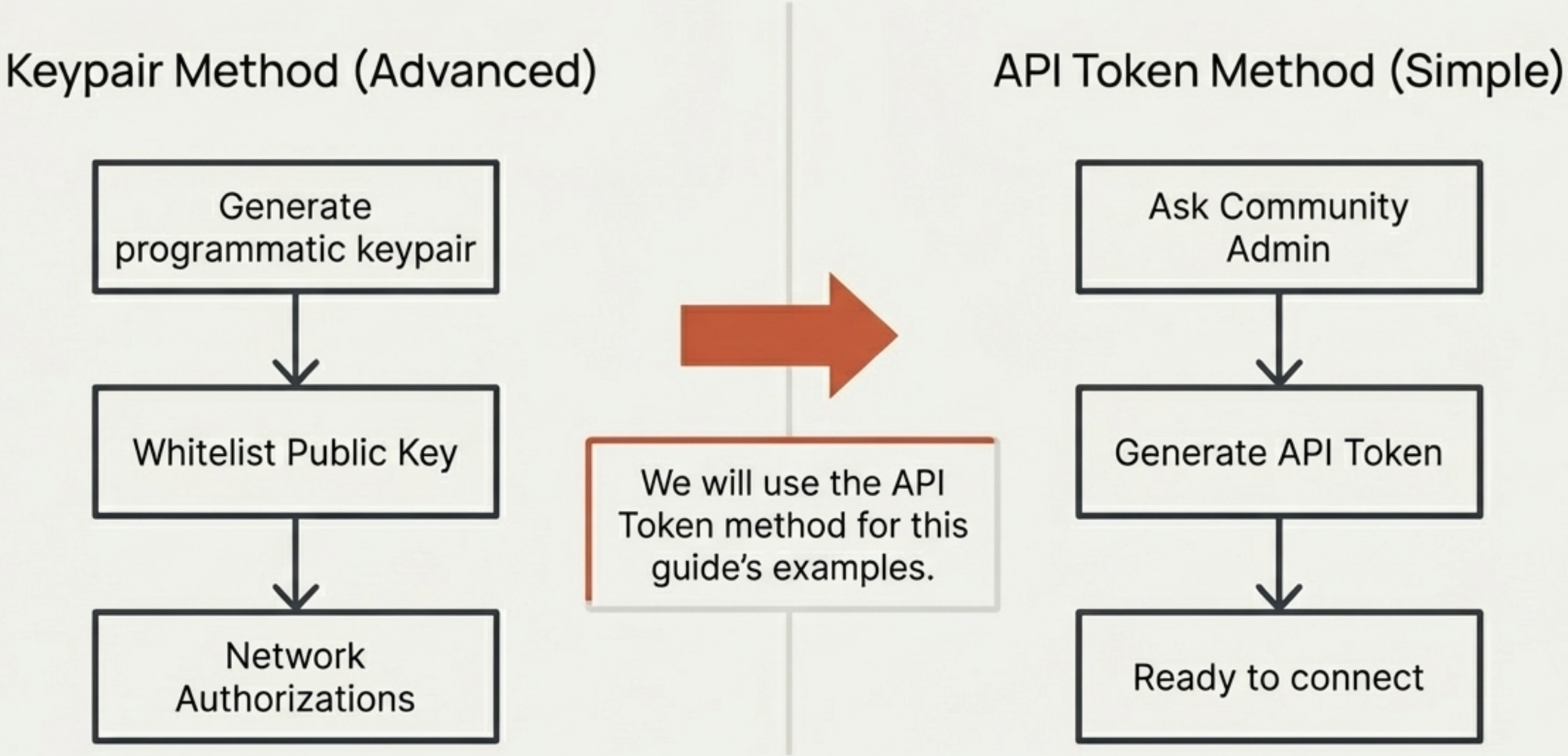
Basic familiarity with Ed25519 keypairs.



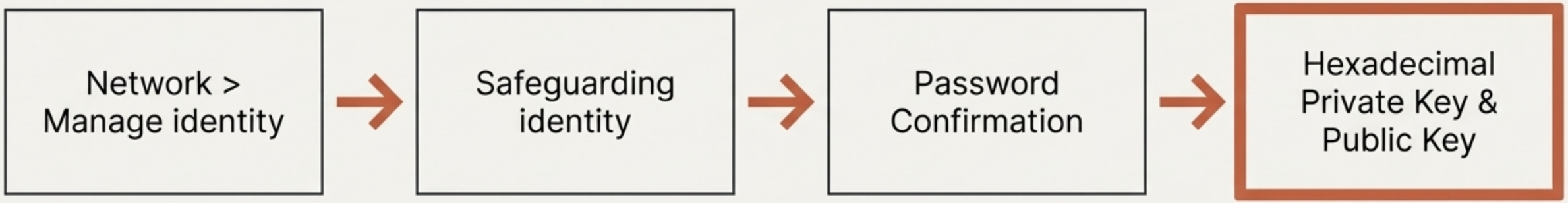
Affinity with an existing Arkipel community.

The Integration Roadmap

Step 1 - Choose Your Authorization Path



Securing Your Community Keypair



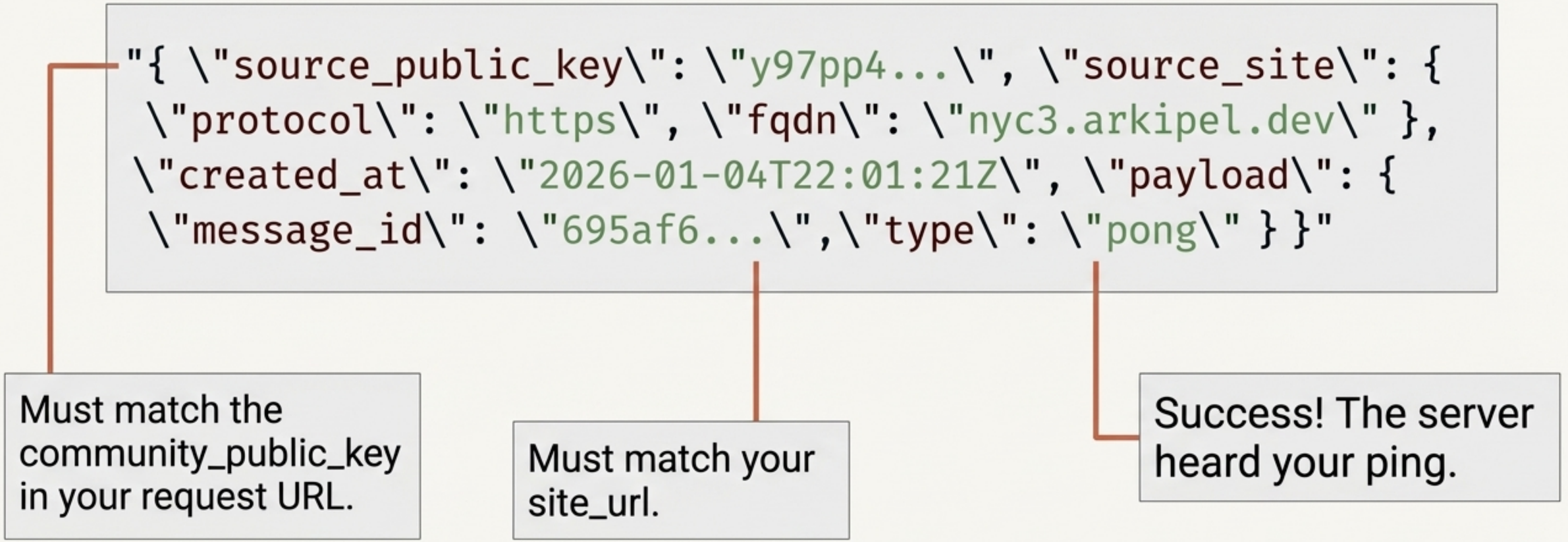
Did you know? Every new Arkipel community automatically generates a keypair upon creation.

Step 2 - The Handshake

Send a simple message with the type `ping` targeting the `/streams` endpoint to verify connectivity.

```
curl -X POST "#{site_url}/arkipel/#{community_public_key}/streams" \
  -H "Authorization: Bearer YOUR_API_TOKEN_HERE" \
  -H "Content-Type: application/json" \
  -d '{"payload": {"type": "ping"}}'
```

Verifying the Response



Step 3 - Creating an Object

Let’s create a person using the people:upsert message type.

We will add Jane Doe to the community.

```
curl -X POST "#{site_url}/arkipel/#{community_public_key}/
streams" \
  -H "Authorization: Bearer YOUR_API_TOKEN_HERE" \
  -H "Content-Type: application/json" \
  -d '{"payload": {
       "type": "people:upsert",
       "first_name": "Jane",
       "last_name": "Doe",
       "import_id": "MY_PERSON_ID"}}'
```

The Asynchronous Catch

```
"payload": {
  "message_id": "695af89d74db9c0a888d286a",
  "type": "people:upsert"
}
```

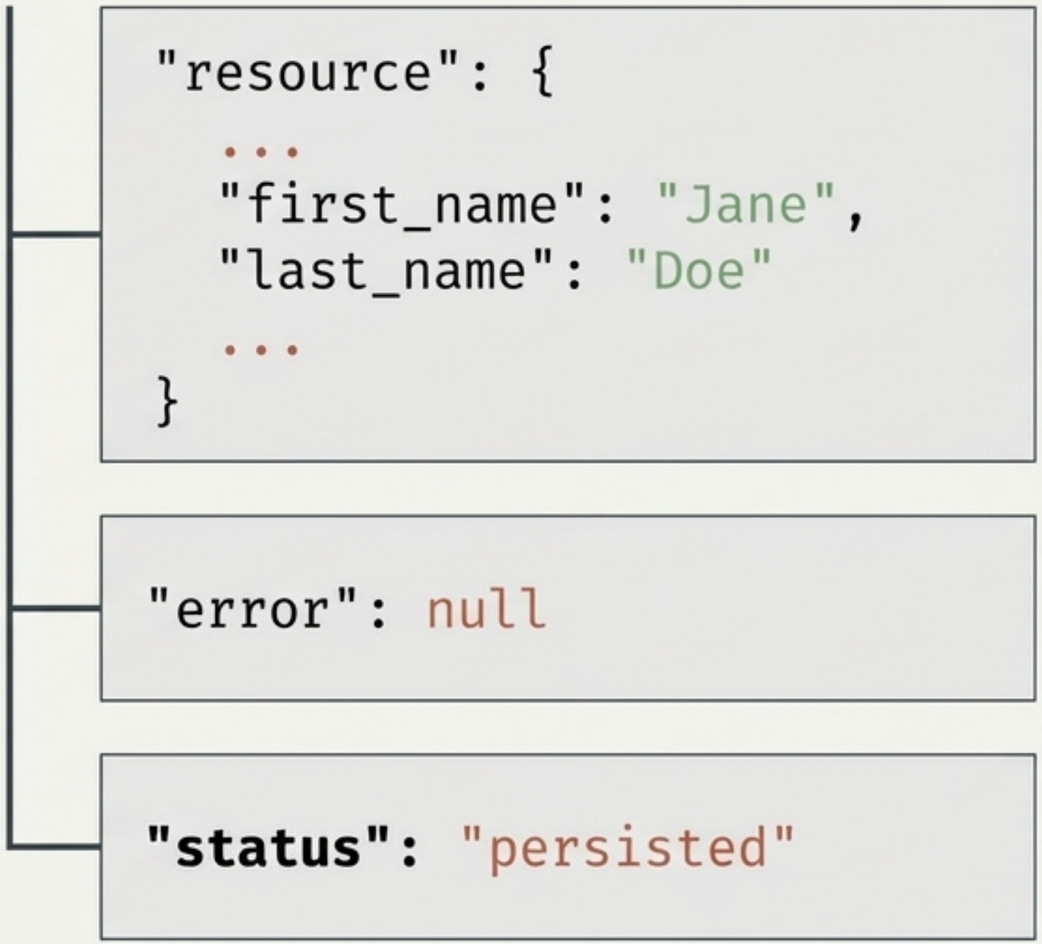
Arkipel processes messages asynchronously. Save this `message_id`—you will need it to check if your upsert was ultimately successful.

Step 4 - Checking Message Status

Use the arkipel_messages:query type along with your saved message_id to verify the system processed your request.

```
curl -X POST
"#{site_url}/arkipel/#{community_public_k
key}/streams" \
  -H "Authorization: Bearer
YOUR_API_TOKEN_HERE" \
  -H "Content-Type: application/json" \
  -d '{"payload":
    {"type": "arkipel_messages:query",
     "message_id":
     "695af89d74db9c0a888d286a"
}}'
```

Confirming Persistence



Contains all available attributes of the created person.

Description of the last error, if any.

The ultimate success `persited`. The community accepted and saved your upsert.

Step 5 - Querying the Community

To finish, let's explore person_categories:query.

We can pass search parameters, such as name_cont (name contains), to find specific records.

```
curl -X POST \
 "#{site_url}/arkipel/#{community_pub
lic_key}/streams" \
 "-H "Authorization: Bearer
 YOUR_API_TOKEN_HERE" \
 "-H "Content-Type: application/json"
 "-d '{"payload": {
   {"type": "person_categories:query",
   "q": {
     "name_cont": "REPLACE_ME..."
   }
 } }}'
```

Navigating Collections

Block 1: The Metadata

```
"q": {
  "per_page": 20,
  "page": 1,
  "total": 2
}
```

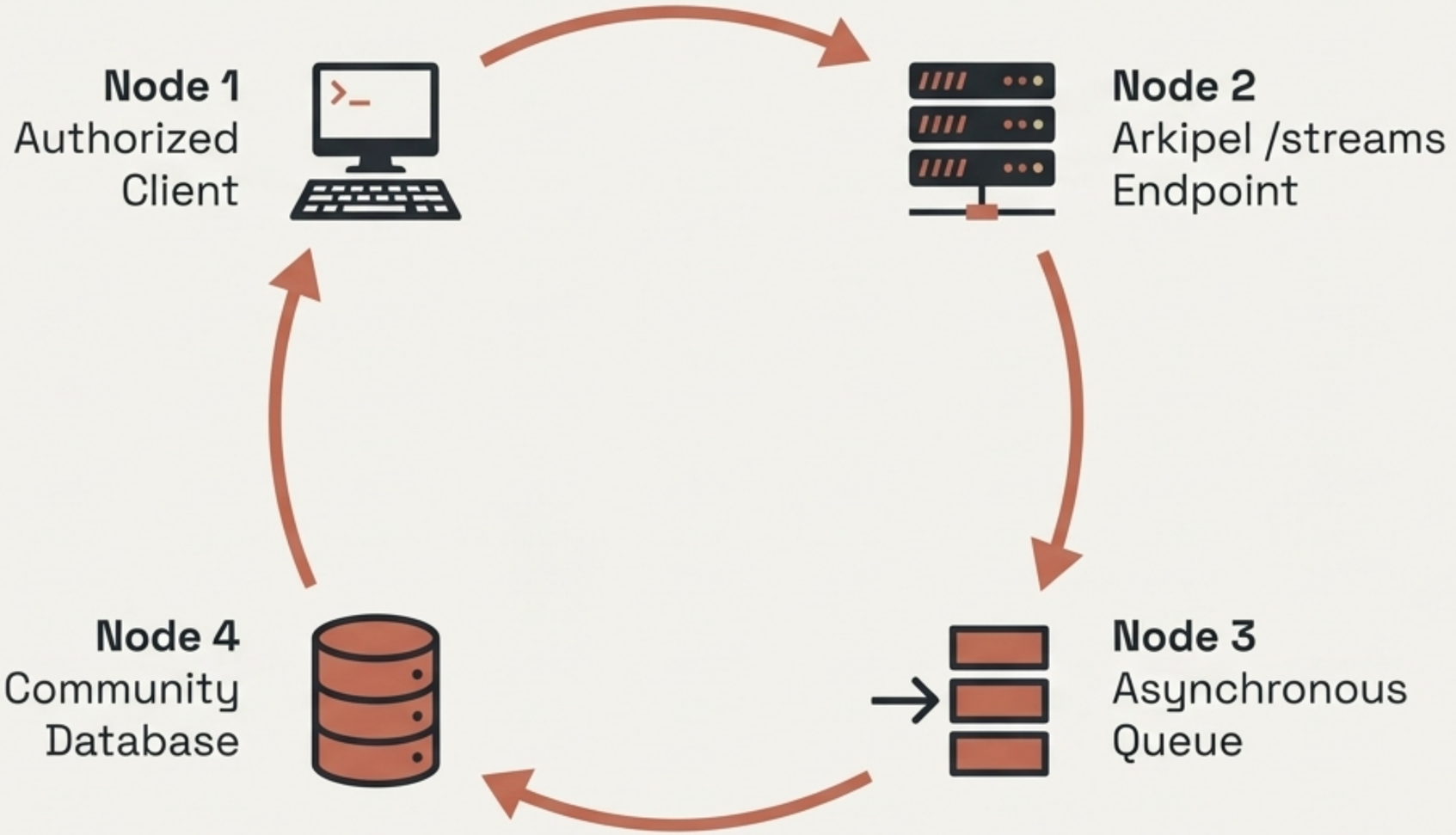
Explains per_page, page, and total pagination context.

Block 2: The Data

```
"resources": [
  {
    "id": 553,
    "name_en": "Family"
  },
  ...
]
```

The array of actual records matching the query string.

The Complete Architecture Flow



You Are Ready to Build

Your connection is live. Expand your integration using the core documentation.

Arkipel Message Specifications

Keypair Generation Document