API Key Management Findings

Kyle Fiedler | Product Designer | kyle.fiedler@consensys.net

Background

We have different key types for different APIs. Some are in the Eth1 space, others are in unified key spaces. We've seen slow adoption for new chains compared to the adoption of Ethereum. We're also introducing new L1s like Near and there isn't a clear flow for adding project IDs to utilize the chain. We don't have a good idea how developers on Infura think about organizing project IDs, in relation to their Dapps, networks, and chains. This research seeks to better understand users' mental model of API IDs and how they would like their IDs to be organized.

Approach

We used Looker to identify ideal users who had multiple projects on multiple networks. After identifying those users, we conducted interviews with 3 participants over the course of 1 weeks. One participant was building an AMM, one participant was using Infura for personal projects creating bots, and one participant was creating a stable coin.

Research Questions

- How are people currently organizing their project IDs?
- How do users use API keys for their project?
- What do users think the relationship is between project IDs and network chains in a Dapp?
- Do they think that their current organization method is effective?
- How are users currently using L2 networks in their development process?
- When do they determine that they need to go multi-network? Multi-chain?

Summary of Findings

1. We found that users organize their project IDs in 4 main ways:
    a. A key for each development environment
    b. One key for all chains and networks
    c. A key for each network
    d. A key for each component of the application
2. All participants thought that the way that they were currently organizing their keyspace was effective.
3. Users find chains and learn about them through several different mediums. The three main reasons were:
    a. Cost
    b. Performance
    c. Access

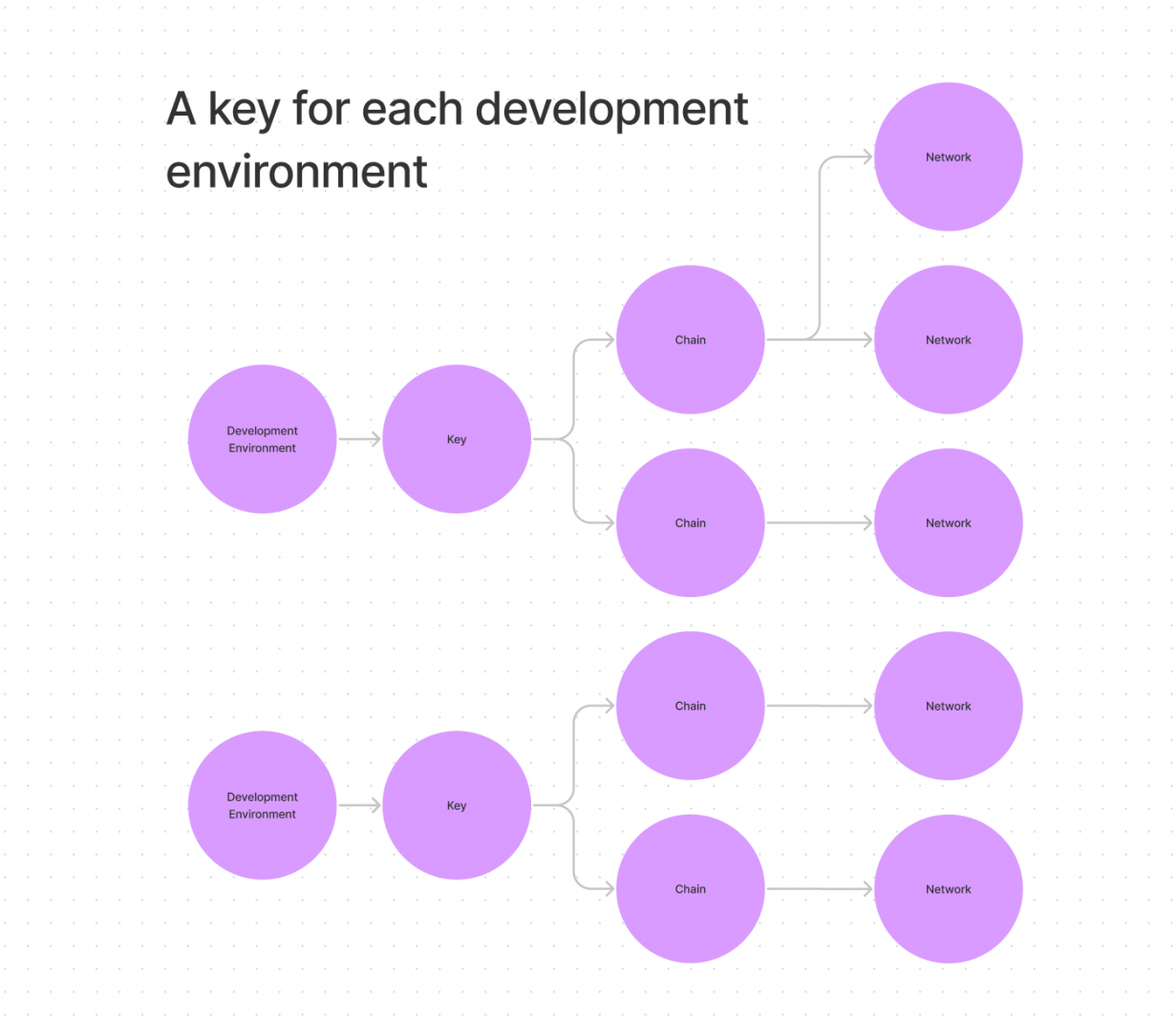
Detailed Research Findings

How are people currently organizing their project IDs? How do users use API keys for their project? What do users think the relationship is between project IDs and network chains in a Dapp?

We found users organize their project IDs in 4 main ways:

1. A key for each development environment

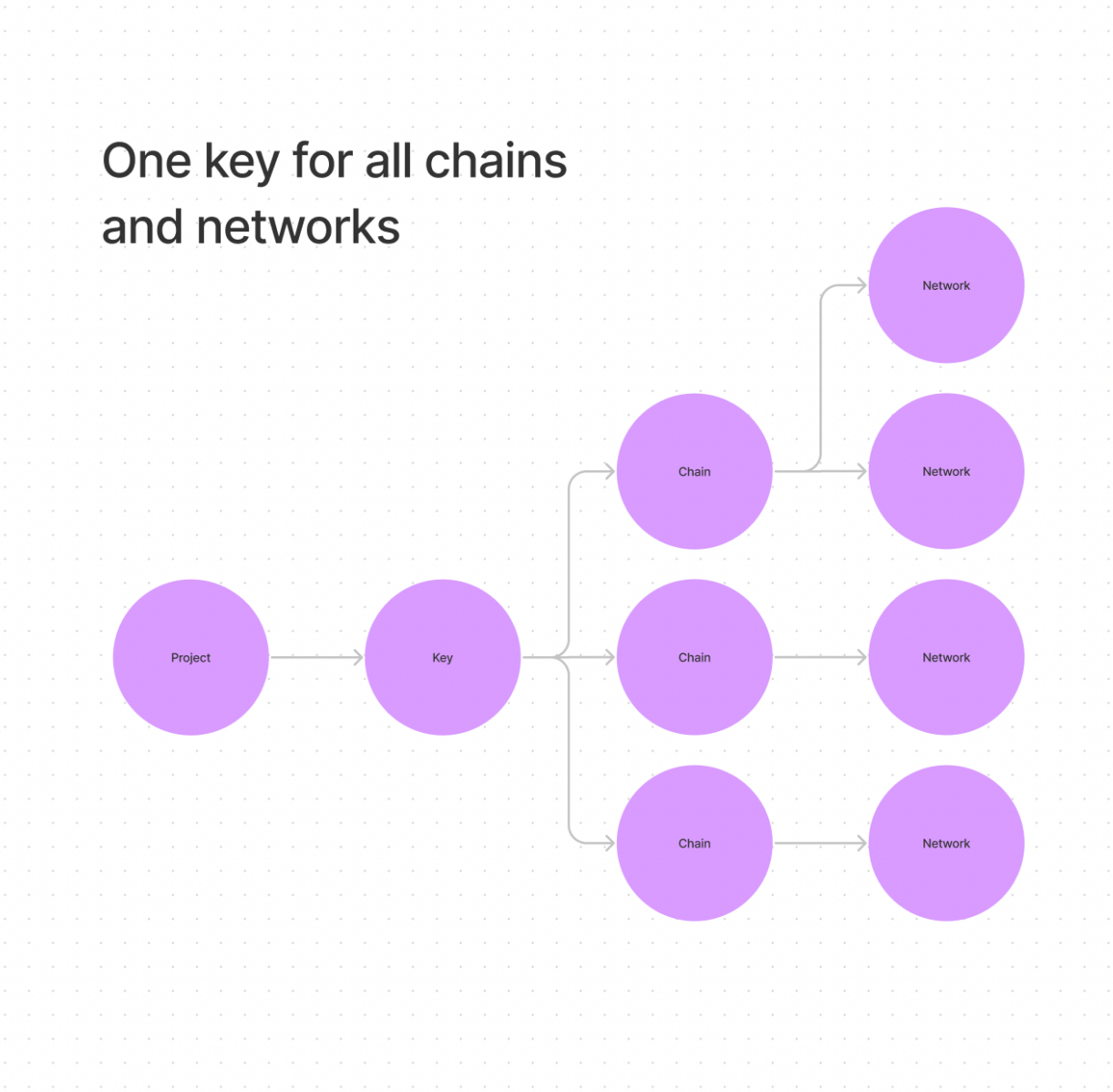
These users followed the Web2 practice of having a key for development, staging, and production so that developers can mitigate potential risks like compromised key or coming close to request limits. These keys would each be used across multiple chains and networks for each of the environments. If one key is compromised, developers can quickly turn it off and the other project IDs aren't impacted. Developers will also track the requests usage and make adjustments to how many requests each key sends. For example, they might notice that they're approaching their daily limit and stop requests from development and staging keys so that production doesn't get rate limited.



2. One key for all chains and networks

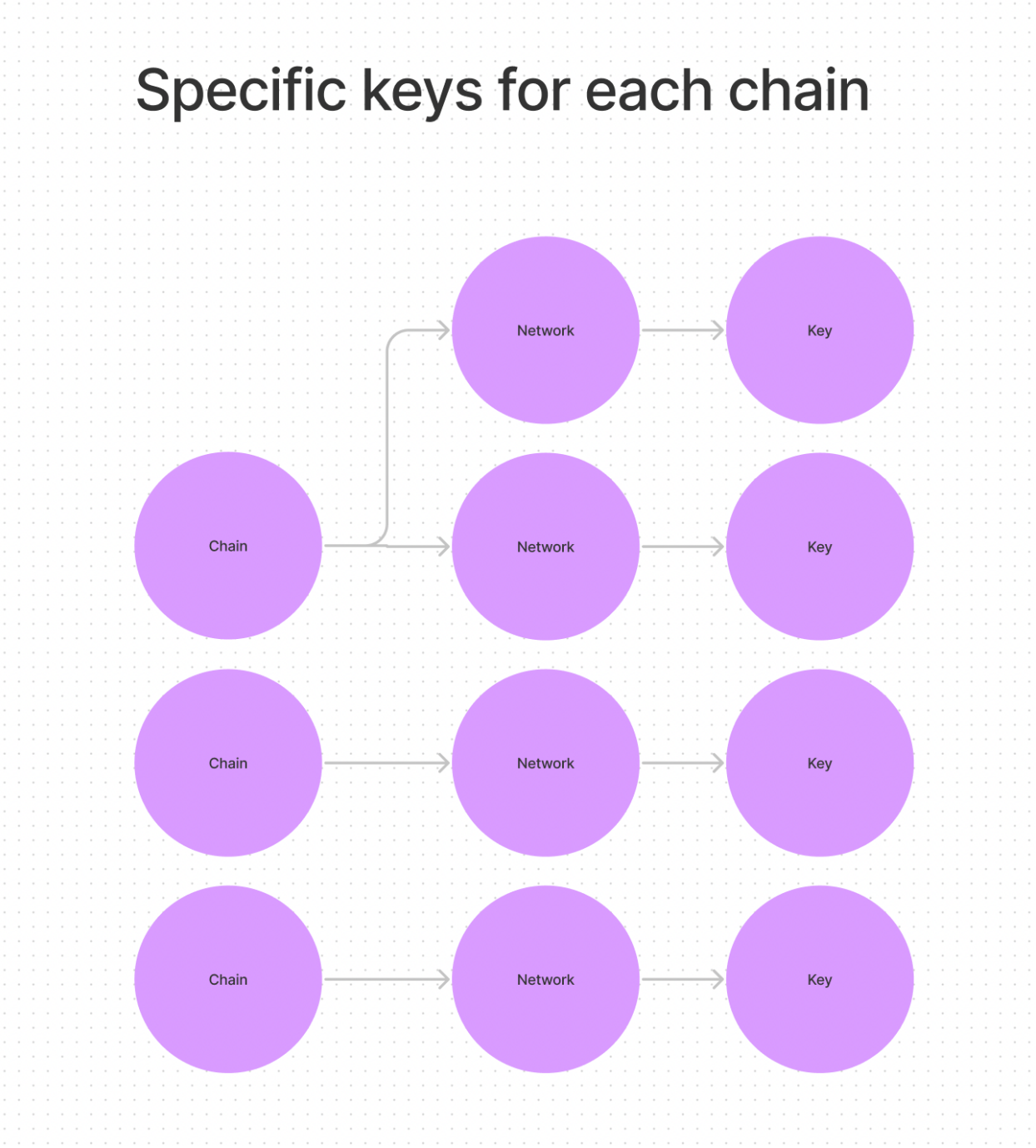
These users use one key for each of the blockchains and networks that they are getting responses from. They like the simplicity in being able to use one API Key for all the networks. These users are most likely to be experimenting and solo developers where keeping things relatively simple is important. They don't need to worry about multiple keyspaces because the applications that they're building are closer to a playground than a heavily used production level application. It seems that users who are building bots are more likely to use one key for all chains and networks, or a different key for each bot they build.

> "I like that I can set up one API Key that's useful for different networks. Instead of having 10 different API keys, but they're all the same application, right, I really like the simplicity there." –P1



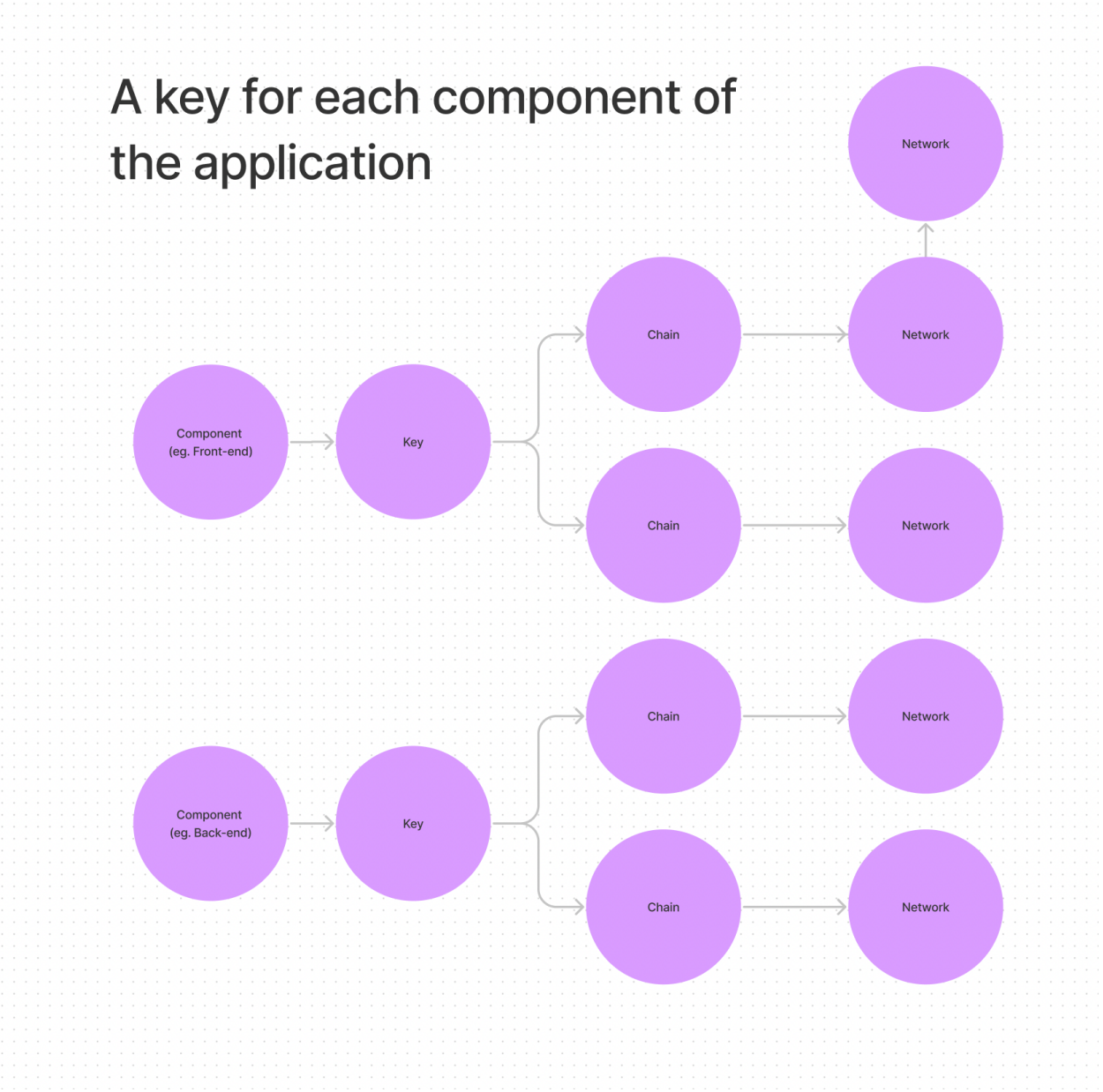
3. A key for each network

For these users, they create a new key for each network that they're building on. They see the relationship between project, keys, and networks as 1 to 1 to 1. This allows them to create a strong structure to their project organization. This is so that the developer can track how many requests are coming through each network through the stats dashboard. This allows them to make product and development decisions based on network usage.



4. A key for each component of the application

These users create a project ID associated with a different component of the app or service they are building so that the developer can track the usage of each of the different components in the stats dashboard. For example front end, smart contract deployment, back end. Smart contract deployments only happen occasionally so they'll have a spike of requests but normally not have any. Developers can easily filter the information from those spikes by separating the smart contract into its own project ID.



Do they think that their current organization method is effective?

All participants thought that the way that they were currently organizing their keyspace was effective. None could think of better ways to organize it for their project.

> “There is no other way, yea, it’s the most effective. I don’t know what would be another use case.” - P3

How are users currently using L2 networks in their development process? When do they determine that they need to go multi-network? Multi-chain?

Users find chains and learn about them through several different mediums. Most couldn't remember how they heard about the ones that they were utilizing though. There were three drivers for going to a new chain:

Cost

Several participants found that building off of Ethereum was more costly and they were looking for different ways to reduce. They saw L2s as a way to significantly reduce cost without compromising on other factors. Some participants saw the L2s as a way to test out production code without having to pay for the opportunity cost of running it on Ethereum mainnet.

> "I play around on the cheap ones and then I'll put them on Ethereum"– P2

Speed and performance

All participants mentioned performance increases when moving to an L2. They mentioned that the best L2 were the ones that not as many people were expecting responses from. But one participant mentioned that once they saw an increase in usage on the L2 they began to see performance slip and wanted to move to another network again.

> "Arbitrum is the best right now, cheap transitions, Optimism is right behind, both transactions finish near instantly" – P2

Access to specific Dapps, tokens or swaps

A key driver for a few participants was access to specific Dapps, tokens, or swaps. For one participant, this was to identify new opportunities for investment, track existing investments, and continually calculate risk adjusted returns. For another participant, they were looking for access to Uniswap through Polygon.

> "That's because the next integrations we were doing is on perpetual protocol and perpetual protocol decided to be Optimism." –P3

Next Steps

- [ ] Tom and Kyle meet to work through user workflow and network organization
- [ ] Tom and Kyle present research and solution to the team. Workshop through their feedback and opinions to get whole teamODD buy in.