GRAPH CHAMPIONS GUIDE

APOLLO

Welcome Graph Champions!

Over the last dozen years we've witnessed the rise of app stores, cloud-native services, agile development, reactive front-ends, and now GraphQL. As the rate of change accelerates, the challenge becomes deciding when to adopt new technology, for what purpose, and how quickly. Internal champions are often the answer, using their foresight and influence to shape an organization's technology adoption.

This is especially true for GraphQL. At Apollo, we see the data graph as a first-order innovation on par with cloud-native infrastructure and the modern app because it represents a new composable API layer for decoupled collaboration. Yet the promise of the graph would remain unrealized without internal advocates to champion the way forward. They see the promise early on, are driven to see it come to life, and work every day to build consensus, generate momentum, and most importantly demonstrate the transformative power of the graph.

If you and your teams see the promise of the graph but are unsure how to start, or get over the next hurdle, read on, this multi-part guide is for you. Through several chapters, we explore the entire graph journey from initial concept to consensus building, offering practical tips you can apply to ensure your organization realizes the full potential of the graph.

A Schema For This Guide

This guide is constructed as a series of chapters that follows the typical graph adoption journey. *Part One: Starting out* begins with a review of the champion mindset, then offers a few grounding questions about your organization to help guide your next steps. Next we discuss how to orient your graph adoption plans and build and effective pitch around the key objectives of your company. Finally we walk through a go-to-market plan, review different considerations when building a proof of concept (POC), and discuss the first demonstration of value. In the subsequent parts of this series, we will discuss issues that arise when scaling from one team to one graph and explore how to keep the momentum going in the face of technical, organizational, and political challenges.

Part one: Starting out

Whether you're at the start of the journey or already on your way, it's helpful to begin with first principles. Then we'll share practical advice for defining your objectives, aligning your efforts to your larger company goals, creating consensus with pitches, working toward your first proof of concept, and demonstrating the value of the graph.

The Champion Mindset

Championing a cause or a technology isn't a role you hold, it's an attitude and behavior you practice.

Anyone who sees the potential of a common graph and feels the responsibility to help realize that promise can be a graph champion. The most effective champions use a combination of skills and techniques along the way, including:

- **Vision** - Champions are among the first within their organizations to realize the potential of a common graph to address long-lived challenges with their current API strategy and app development approach. They often use terms like a "single sheet of glass" or a "common source of truth" and tend to focus on the larger organizational impacts of a common graph versus the narrower technical benefits of GraphQL.
- **Collaboration** - Champions are catalysts who operate within and on behalf of teams. Champions realize large-scale change is a team sport and their first goal is building a small, aligned team with a shared vision and purpose.
- **Persuasion** - Champions are influential within their organizations. They understand how to build a compelling case for leadership, then support it with arguments and data to overcome doubts. The most successful champions understand the importance of translating a common graph's technical benefits into benefits for the business and customer.
- **Agility** - Champions must balance their long-term vision with a willingness to adapt to new learnings and new circumstances. Their classic response to setbacks is introspection, humble adaptation, and new consensus building.

Few hold all of those qualities in equal measure and the purpose of this book is to help every champion regardless of where they are in their journey. Think of this as a cheat-sheet for all graph champions. Let's dive in!

Queries First: Get The Lay Of Your Land

First let's acknowledge the best path to building momentum behind your graph initiative will depend on your organization, its culture, its leadership, and the situation on the ground. Appreciating those local conditions and adjusting your approach to suit can make the difference. To help here are some questions to ask yourself and your team.

DO YOU OPERATE IN A SHOW OR TELL CULTURE?

Does your organization rely on thorough arguments written out with supporting data, or compelling proof points in production? If it's clear, see the sections below on tips for both. If you're not sure, rather than guessing, confirm by asking the key decision-makers a question like this:

> *"My team has been looking into GraphQL and we believe it will address some long-standing challenges impacting our team's velocity and the customer experience. We're ready to make the case for moving in that direction. How would you advise we best present our thinking? A working POC or a pitch deck/doc that lays out our thoughts?"*

Involving your leaders early on, grounding your case in customer and productivity benefits, and seeking their guidance demonstrates you are looking beyond the technical advantages and acting like a champion.

IS THERE A WAVE YOU CAN SURF?

Occasionally internal priorities align and large initiatives arise. Whether triggered by internal or external forces, big internal shifts are a great opportunity because they shake up the status quo. Leaders seeking new outcomes are willing to embrace new approaches. Examples are major refreshes of your product line, migrations to the cloud or new leaders seeking 10x ideas. Sometimes these fall into your lap and they appear as a top-down mandate to invest in becoming a platform company and the connection to GraphQL's strengths is obvious. In other cases, you need to help leaders see how a common graph can accelerate progress toward the new priority. We'll dig into this deeply in the next section.

WHAT DOES SUCCESS LOOK LIKE?

Even if your company's culture encourages teams to explore new approaches without any pre-approval, it's useful to think through the ingredients for big-time success. Who are the key decision-makers or influencers within your group, your org and your larger company? What evidence would you need to collect and present to make a compelling case for transformative value? Like a startup, you should always be thinking about securing your next "round of funding".

WHO'S ON YOUR TEAM?

As we'll discuss in later chapters of this multi-part guide, ensuring the graph as a shared part of your platform infrastructure often ends up being the platform team's responsibility. Aspects like CI/CD integration, localization, authentication, and authorization underscore the importance of having devs on your graph team who are very familiar with your platform. Yet there are other team members who can bring their own key perspectives:

- **Front-end app devs** - While APIs are typically built by service teams, graph schemas work best when shaped for their primary customer—the front-ends. Bringing an app and UX point of view into your team is a great way to gain that perspective.

- **Product managers** - Work on the graph often competes with other priorities. Having a teammate that speaks and thinks from the product perspective can be invaluable when you are asking for an investment tradeoff to be made in favor of the graph.

- **Designers** - Design systems and graph schemas are both examples of systems thinking. Adding experts in information architecture helps guide your schema design efforts. And design leaders can be your strongest advocates for a systematic data and capabilities layer, so including them into your team is a win-win.

- **Communications & Internal Education experts** - When embarking on a mission of change bringing along guides who understand change theory and know how to communicate at scale can be invaluable. Invite them to join your mission!

And finally, if the graph is new to your company, it is often better to start with a virtual team in the beginning, rather than trying to organize a new team within your organization. This lets you pull in people who are passionate about GraphQL quickly, without the cost and delay of forming a standing team.

Tie your graph goals to company objectives

Technical benefits are best appreciated when related to positive outcomes for your team, customers, and business

A good next step is to humbly realize that desires for a better tomorrow powered by GraphQL and a common graph don't exist in a vacuum. Your organization has its own larger objectives and priorities and your leaders will rightly focus on delivering those objectives. A champion's job is to help your leaders see how a common graph helps you reach those objectives and delivers value the leadership team measures and seeks to maximize. Just as leaders think about value delivery in multiple ways, so should champions. Casting the value of the graph in customer-benefit terms your product and design leaders use will make your message more broadly understood by your company. Thinking like your business leaders allows you to tie your message to results that drive the health of the business.

With that in mind, let's review a list of four common objectives we hear. In each case we'll look at the pain points that are in the way, how the graph helps address them and delivers value that matters.

Four common objectives:

- **Faster product development**
- **Improved UX performance**
- **Consistent customer experiences**
- **Escaping the monolith**

OBJECTIVE – FASTER PRODUCT DEVELOPMENT

The day-to-day experience of your front-end and back-end developers directly affects their productivity, effectiveness, and ultimately the rate at which they can build, iterate, and improve your digital products.

Understand the pain points

- With REST APIs, app devs spend a lot of time writing and managing repetitive service API integration code on each and every device platform you support. Managing versioned API endpoints, discovering the right service to call, coordinating with service teams, and working through API-specific semantics means wasted time and effort and developer frustration.

- Service oriented architectures often lead to each service defining their own models for common domain types. It is common to see two, three, or more models for concepts such as products. Clients are left to figure out which is the right one for their use case, and where "right" might be most recently refreshed or most complete. Even when service teams agree on a common model, keeping those models in sync across services is impractical.

Problem solve with the graph

- A common graph is a single place for app devs to discover the data they need, regardless of source. GraphQL's query language and powerful graph schema explorer tools lets them quickly access what they need, try out new ideas, and focus more of their time on building rich customer experiences.
- With federation, service teams can add new capabilities to existing schema types, for example adding reviews to the product and customer types without requiring apps to manage the underlying service orchestration or add new endpoints.

Translate graph benefits to team, business, and customer value

- **Eliminate frustrating and tedious work.** Your best app devs joined to build great experiences, not write boilerplate service integration code. Freeing them up to focus improves productivity and morale. Ask you front-end devs how much of their time is actually spent building UX, you may be saddened to learn it's less than you assumed!
- **Accelerate product iteration.** The quicker devs can try out new product ideas via experimentation, the faster your product evolves and improves. Ask your product leaders what increasing product delivery times is worth to them.
- **Delight customers.** Rich and fluid customer experiences take time to get right. Freeing your best app devs to focus on them should be a priority. Ask your design and usability leaders what their research tells them about the value of great customer experiences.

OBJECTIVE – IMPROVED UX PERFORMANCE

Performance, specifically the end-customer user experience performance is always a key objective. Whether you are building a mobile e-commerce path or an interactive exercise dashboard, avoiding inefficient patterns and providing the observability insights needed to optimize your performance and then keep it fast is a top objective for every company.

Understand the pain points

- REST APIs return fixed responses for every request, so clients often get more than they want but less than they need. Large payloads and multiple sequential round trips are the technical result.
- On the back-end, service teams struggle to meet all their client's needs without duplicating functionality found in other service endpoints or requiring apps to orchestrate across multiple service APIs.

Problem solve with the graph

- Multiple service calls become a single optimized query plan, avoiding over-fetches, under-fetches, and multiple fetches.
- With graph-native observability tools your team can see the complete front-end request and response in one place so you can optimize the end-to-end experience rather than each service endpoint.

Translate graph benefits to team, business, and customer value

- **Increased customer satisfaction and conversion.** The probability of a bounce increases 32% as page load time goes from 1 second to 3 seconds. Walmart found that for every 1 second improvement in page load time, conversions increased by 2%.
- **Greater product insights.** With graph-native observability, product managers can see which clients are using which data and features.
- **Usage insights enable optimizations.** Service teams can spot hotspots and understand which clients and use cases to tune for. Moving service orchestration out of each client makes it visible and reveals potential optimizations.

OBJECTIVE – CONSISTENT CUSTOMER EXPERIENCES

As people use your product, what they learn from one device touchpoint is carried to their expectations of how the next will look and feel. If customers need to re-learn how to perform common tasks across different devices, platforms, or parts of your business, using your product becomes more difficult and negative emotions result. Inconsistent product behaviors also make it difficult for customers to recognize a common brand identity and build trust.

Understand the pain points

- When front-end apps need to implement complex service orchestration they become filled with complex logic. Because that code must be written for each app platform by different teams it's no surprise the customer experiences drift apart, even unintentionally.
- Beyond orchestration work, without a shared layer for clients to find consistent answers, it's very expensive to keep feature and behavior parity across all clients. Especially when some clients earn a small fraction of total engagement or revenue.

Problem solve with the graph

- A graph's abstract schema serves as a consistent contract for product answers. And one that has been defined at a layer independent of both a specific client or service.
- Standardizing logic behind schemas creates consistent behavior. AI/ML models built server-side can bring intelligence and adaptive behavior to all your app experiences.
- If you have a design system, you can explore returning design system tokens that each client understands how to translate into standardized UI elements.

Translate graph benefits to team, business, and customer value

- **Reduce work for devs and deliver better results for customers.** If the graph communicates more of the product behavior while client apps do less, it means there is less code to drift and become fragmented.
- **Boost return rate and engagement to drive revenue.** Consistent experiences reduce cognitive load and lead to more positive emotions, which increases trustworthiness and brings users back.
- **Centrally drive experiences.** Consistent answers lead to more consistent experiences, and integration with your design system is a powerful way to centrally drive experiences in cases where the UX is highly regular and template based.

OBJECTIVE – ESCAPING THE MONOLITH

Large monoliths are literally repositories of millions of hard-fought learnings, winning strategies, and practical workarounds. For companies that rely on them, they are simultaneously a precious resource and a large impediment to velocity and agility. How to safely escape their gravitational pull is a challenge many enterprises face.

Understand the pain points

- By the time monoliths earn their pejorative title they are cumbersome to manage, difficult to change, and yet crucial to the business.
- Breaking them up into microservices is not only a process that takes years, it is fraught with technical risk and major disruptions for the client applications and the customers who use them.
- Yet pressure to address the elephant in the stack only grows with time and the needs of the business.

Problem solve with the graph

- The graph's ability to model a cohesive set of data and capabilities with an abstract schema before that model exists at the service layer is a uniquely powerful tool for safely and iteratively migrating away from your monolith. Once you've insulated your clients from your monolith via a graph, you are free to break up your monolith in a piecemeal fashion, on a schedule that decouples the front-end and back-end work.
- Sometimes monoliths aren't born in house, but acquired through an external acquisition or merger. In those cases, the same graph-first migration pattern can help deal with the complexity of multiple platforms.

Translate graph benefits to team, business, and customer value

- **Productivity remains high.** Once client apps move to the graph, they become insulated from the impact of breaking up the monolith. That decoupling means front-end teams can continue to work on their highest-priority work, safe in the knowledge that the stable functionality of the monolith is preserved, albeit now exposed through a much nicer composable graph API.
- **Continuity for customers.** Customers are less likely to lose functionality or see flaws during the MVP phase of the microservice equivalents.

- **Less business risk, more flexibility.** More broadly, a decoupled migration strategy that captures the core capabilities of the monolith in an abstract schema means less risk to the business and more flexibility about the timing of the monolith breakup into microservices.

Charting your path to first rollout

In this section, we'll explore how to build consensus and then start learning and demonstrating value by delivering a first graph to your customers.

BUILD AN EFFECTIVE PITCH

For your idea to grow strong enough to be widely adopted, it needs to spread and live in more minds than your own. Sometimes you'll make your case via a formal deck, presentation, or document, but regardless there will come a time when you need to build a larger consensus. Effective pitching is something most of us learn by trial and lots of error. Here are some cheat codes to keep in mind whether you're pitching friends, allies, frenemies, or leaders.

1. **Tailor to your audience** - Remember, this pitch isn't for you, it's for a different audience. Think about their context, priorities, pain points, objectives, and the amount of time they can spend listening to you. Assume the answers are different for different constituents and tailor your pitch accordingly.

2. **Practice it often** - When you share your case with someone, watch their reaction closely—what did they like and what made them pause? Each pitch is a chance to refine and improve the idea. Your job isn't to convince them, your job is to see if the idea infects them and if not, why not.

3. **Seek a contrarian view** - Contrary to what you might think, your best friend at this stage is a contrary opinion. Find someone who by nature and relationship won't agree with you to make you feel good or be nice, yet is open enough to new ideas. Someone who sees the world from a different vantage point that will highlight your idea's qualities. When you are pitching them, watch their reaction closely. What do they see that you missed? Are they not buying the premise or the conclusion? Ask them how they would make the idea better. Anything you can glean from this stage is gold, you can prune a weak branch off your idea, graft on a new one of theirs, or hybridize both.

4. **Presell to allies and amplifiers** - Every idea needs help to grow and spread. So beyond practicing your pitch with peers, identifying potential allies who can amplify your message is a great way to increase the chance of success. If you know folks in the product and design orgs

they can be powerful allies. It’s highly likely the head of design and head of product are keenly interested in faster product velocity and more consistent customer experiences. Don’t be afraid to ask for 30 minutes to pitch them on how the graph can help achieve those goals.

5. **Learn to hear “no” as “not yet”** - Good ideas like a common graph require the right conditions to flourish. Sometimes companies are distracted, or your idea finds a strong opponent, whether for idealogic, political, or perfectly sound reasons. Here's an open secret: things change. Leaders change, organizations reorg, market conditions change. If an idea is good, it will have its day, and if you are prepared you can seize the moment and convert a previous no to a big yes.

Go-to-market: Delivering Your Graph Solution To Customers

If your pitch has been greenlit, or you have the autonomy to move forward, it's time to move from tell to show. That means delivering a real graph solution to production. Until you do that, you haven't delivered any customer, team, or business value. As described above, without value delivery, better technology spreads slowly or not at all. But hold on—didn't we skip the POC stage—don't we need to validate GraphQL as an approach and technology before we commit to solving a real-world problem? Absolutely. There's no question teams need experience learning any new technology. Yet whether the learning is best done within a very narrowly defined POC or tied to a more ambitious first goal depends on your situation.

A good argument can be made that solving a real-world problem is the best way to learn, prove out a technology, and define your reference graph infrastructure. So as you start your initial explorations, keep your eye on the real prize: a demonstration of value. Consider whether you can validate the concept while solving a real-world problem that matters to your company. Technical learnings are only one consideration when pressure testing a common graph. Choosing a non-trivial first deliverable lets your teams gain experience on coordinating schema design across teams while also working through ownership, education, and community building.

Let's now walk through three key next steps: choosing your first use case, defining your success criteria, and looking ahead to your reference graph platform. Let's take each of those in turn.

CHOOSE YOUR FIRST USE CASE

If you are a single team and own a single application or service, where to start is pretty clear - start with what you know. But if you have choices, and often you do, here are some considerations that can help your selection:

- **Big bet or play it safe?** Balancing risk vs opportunity is one of the most important decisions. Traditionally, teams seek to minimize risk by choosing a small or well understood problem when attempting something new—and for good reason. Yet if your business is attempting something that you know can't be done with traditional APIs—like rolling out a new customer experience to all your clients in less than one year—moving cautiously and hedging your bets can actually be the riskier play. Specifically, we've worked with a number of large enterprises that needed to make a big move, a transformational product rebuild with no time to spare. Whether its increased competition or the acceleration of trends from a global pandemic, sometimes the bold and decisive all-in approach is the only hand that can win, so going big is safer than hedging your bets.

- **Native first or Web first?** When choosing your first client app, the implications of GraphQL's backward compatibility tilt the board toward Web and away from Native. Why? Native apps are installed onto devices and forcing your customers to upgrade their binaries is difficult or undesirable. Waiting for those app versions to age out may take years. On the other hand, with a web app you can easily deprecate and replace large-scale schema changes at will. In the early days of schema design, everyone is still learning so big changes occur frequently and large schema depreciations may result in schema debt. To be clear, if the business opportunity for native first is there, do it, but if all other things are equal, choosing a web app as your first client is a more forgiving choice.

DEFINING YOUR SUCCESS METRICS

As you prepare your early production rollouts, think back to the objectives and pain points you identified earlier. Use those to define success criteria that will measure progress against those goals. Some of the positive impacts take a while to show up so consider the results outlined below, which are leading and trailing indicators of progress.

Objective	Key results and metrics
Developer productivity	Increased delivery velocity is the gold standard, but don't underestimate the power of anecdotal stories of reduced friction and developer happiness in the early stages.
Consistent user experience	While it's difficult to quantify consistency and cohesiveness, you know it when you see it. Some companies have run usability studies before and after to measure, but others simply keep track of the experience before starting, then compare it to see the improvements and measure other engagement factors known to track with CX improvements.
UX performance	Reduction in payload response, render time, and the number of request/response pairs are key metrics.
Adoption & engagement	Increasing the number of developers and graph clients querying the graph demonstrates growing adoption and engagement.
Adoption & engagement	Increasing the number of services and service teams integrated into the graph reveals your graph growth rate.
Graph health	Successful teams make frequent schema changes. Measuring your schema change rate over time is a good proxy for graph health.
Graph health	Another important metric is the number of breaking API changes. If you are leveraging automated schema checks that number should go down

	over time and that is directly tied to decreased revenue loss and increased customer satisfaction.

LOOK AHEAD TO YOUR REFERENCE GRAPH INFRASTRUCTURE

Like your CI/CD and source-code control systems, there are advantages to standardizing on key parts of your graph infrastructure. Here's a checklist of considerations and decisions to make as you look ahead, past your initial proof of concept.

- **GraphQL clients** - While you can POST GraphQL queries directly to the server, almost all teams enjoy the benefit a GraphQL client like Apollo Client offers. Part of the learning curve is exploring the different clients for each device platform and defining your client caching strategy.
- **Federated GraphQL Server/Gateway** - Federation is a powerful way to create order from distributed chaos without coupling your teams. While your initial graph may only expose a schema based on one or two services it's not too soon to think ahead. Federation allows you to start with a single schema, move to simple composition by combining multiple schemas created and managed by different teams, then later explore ways to link and extend types across service boundaries. Managed federation handles the composition, checks and deploy of a composed schema and ensures your gateways never have to restart to pickup schema changes.
- **Schema Management & Schema Checks** - As your single schema file owned by a single team becomes many--owned by distributed teams and changing daily—how will you be sure that changes are safe and backward compatible? Schema checks are the answer. Static checks at build time are a required part of any production-grade graph infrastructure. But you can level-up by leveraging Apollo Studio's github integration to catch breaking changes at PR time, preventing build breaks before they happen. Apollo Studio's Trace Warehouse also allows you to go beyond static checks to actually validate your changes against the last N days of real world queries. While backward compatibility forever might seem the ideal goal, in practice retiring schema debt safely when an affected client's query volume has dipped below your threshold is a great way to prune your long deprecated portions of your schema.
- **Tracing and Observability** - You'll want to think through how tracing will work from clients through the graph servers and underlying services. End-to-end tracing is a powerful debugging and analysis tool. Check out Apollo's Studio robust graph-native offering in this area.

- **GraphQL Service Chassis** - A number of enterprises have chosen to standardize on a common service chassis that addresses many of these topics in a standard fashion and lowers the effort required to expose a set of domain services or data sources via the graph.

If that seems like a long list, consider two observations. First, thinking ahead is cheap and best done at every stage. Second, while it's prudent for every organization to validate GraphQL and your graph infrastructure, there is a large body of evidence showing how a common graph solves long-standing problems, exceedingly well. In a competitive landscape of dynamic customer expectations, shrinking moats, and increased competition, the smart bet is to move through the validation stage as quickly as possible, ahead of your competition rather than behind.

Wrap-up and takeaways

In this first chapter of our graph champions guide, we covered the principles of being a champion, outlined ways to align graph benefits with your company's broader objectives, and explained how to build consensus through a pitch and/or POC. Key takeaways included:

- The champion mindset is not simply a role you adopt, it's an attitude you practice. The most effective graph champions utilize vision, collaboration, persuasion, and agility to drive their initiatives.
- Look closely at your organization to assess the best path forward for graph adoption. This could mean beginning with a pitch and a POC, aligning with a larger organizational movement, bringing along key decision makers, or broadening your graph team.
- Before jumping into a pitch or a POC, think of the key objectives your company wants to achieve. From there, determine the pain points that stand in the way of that goal, how the graph solves those problems, and how the graph enables value for your team, customers, and the business.
- And finally, build consensus and demonstrate value through a pitch and/or a POC. For a pitch, make sure it is pointed, practiced, and supported by allies. For your POC, first choose your use case, then define your success metrics, and lastly validate your reference graph infrastructure.

But this is only the beginning of your graph journey! Stay tuned for the following sections of this guide that explore how to scale your graph initiative and maintain momentum in the face of potential issues.