okta

Driving Innovation with Architectural Excellence

Okta Inc.
100 First Street
San Francisco, CA 94105

info@okta.com
1-888-722-7871

Introduction

Today's organizations are under a lot of pressure. Not only do they need to provide innovative technology solutions, but they also need to ensure these solutions evolve with changing customer needs—and that means having excellent architecture. However, in the neverending push to develop new, marketable product features, many companies don't pay enough attention to their technology architecture, a strategy that hurts them in the long run.

Allocating ongoing resources to enhance your technology stack and software architecture will pay your team back with time to innovate on your products. But what does this look like in practice? First, it means taking a periodic look at your architecture and ensuring that it represents the right long-term foundation. This review should include both the technologies you stand on as well as your own intellectual property. Second, it means transforming technical debt into an asset.

Evolving alongside current infrastructure can help your business innovate at scale, extending beyond the original scope of your product or service. This is particularly true with cloud services that are rapidly advancing their scaling and functional capabilities. The existing foundation is constantly moving, affording an opportunity for companies to stand on higher ground. But the cost of change is high, and architects don't have the time and resources to chase every shiny new project. Striking the right balance here means finding a middle ground between being too aggressive and constantly churning your infrastructure, or moving too slowly and being stuck with proprietary building blocks.

Similarly, within your own software, the languages and tools need continual reevaluation—that way, they can remain modern, without being fragmented into too many dialects and frameworks. Additionally, in a more subtle approach, the major modules of code should be reviewed. Consider this: does a critical subsystem warrant a major refactor because it is an impediment to innovation? By re-writing and perhaps combining code across teams, can you make it easier for your product suite to advance into new markets? The right technical debt paydown can turn into product equity and enterprise value.

Of course, following this approach will require technical teams to change some of their thinking and processes. Here's how your organization can focus on architecture while continuing to delight your customers with new products and services.

Where architectural excellence and innovation meet

Building software is like writing a giant novel: the effort is part craft, part science, and part art. Unlike novels, however, large applications are developed by scores of contributors. As your company grows, simply pouring more money and staff into your process isn't always the best solution—this has been well known for decades. In order to deliver robust and agile software, your team members need to be organized with the right infrastructure and architectural choices.

Principle #1: Explore emergent properties

It's often said that the whole can be greater than the sum of its parts—and the same can be said about well-designed software platforms.

Amazon Web Services (AWS) is the canonical example here. After developing numerous internal tools to maintain their storage, computing, and networking resources, AWS created a new revenue stream by bundling these tools and renting them to third-party vendors. What emerged was an entirely new business category that enabled new companies and industries to flourish as a result of the emergent properties of pay-as-you-go infrastructure and the virtuous cycle of new services with low friction.

Consider the Okta Integration Network and Okta Identity Engine. These are examples of complex

systems that were born from simpler beginnings: single sign-on (SSO) and identity protocol integrations. When these pieces were brought together, there was suddenly a new way to build applications that connected businesses and customers securely. After first focusing on employee identities, Okta adapted this same infrastructure and innovation to partner portals. This was followed by consumer and marketplace identity solutions, which allowed for more innovation.

By iterating on and generalizing those core primitives, Okta's platform has expanded from simply connecting SaaS applications to handling any manner of hybrid clouds (including on-premises directories and authorization agents). It has also grown from being based primarily on browser-based applications to now delivering programmable API-first identity solutions.

For an organization to capitalize on similar emergent properties it needs to evaluate the core services it does well, and then iterate and polish them to a point where they can be reused and composed across a variety of use cases—or applied to entirely new ones. This is something that's happening across the industry. Some of the strongest platform network effects today were born from refining and iterating on simple concrete product offerings.

Principle #2: Enhance engineering productivity

Ensuring your developers are well-equipped to build efficiently is crucial to your success. But if your product is incapable of scaling or weighed down by brittle code, then your teams won't be able to achieve their full potential. Instead of focusing on innovation, your developers will spend significant time fixing issues or building workarounds for the current architecture.

Tend your garden

It takes a lot of human effort to fight entropy and achieve order and polish. Like maintaining a garden, keeping software organized and beautiful requires a steady vigilance. Once an area of code develops enough weeds, then it becomes harder to justify—or execute—elegant fixes. Developers are more likely to "band-aid" that area instead of improving it systemically. Eventually it falls into disrepair, and no one wants to touch it. Similarly, if a feature misses the mark and lacks functional polish, then it won't work well with other parts of your platform. Product managers, knowing that the feature is hard to evolve, will let it languish and accumulate more rough edges. To mitigate this, you should encourage practices that ensure your team's code is responsive to both end-users and developers alike.

In addition to embracing efficient coding processes like continuous integration, testing and delivery, leading technology companies mitigate the threat of code deterioration by monitoring their infrastructure for bottlenecks and problem areas before they escalate. Developers tend to know where the issues are in the code. Empower your technical leaders to speak up and act to prevent important modules from becoming graveyards.

Evaluate bigger change

Beyond evaluating the code for fit and finish, sometimes a larger surgery is required. When organizations look to evolve their platform or product, they're likely to encounter modules that need a wholesale rewrite or refactor. This may be because that component will have to assume more importance or scale beyond its initial implementation.

Executing a component rewrite is no small task, but it can be done with careful planning. Often, the old and new modules are run in parallel with the new system—first running passively alongside the previous one to make sure they behave the same. Within this process, the cutover can be done in a controlled fashion, starting with a small percentage of customers, for instance. Eventually, the kinks are ironed out and the previous solution is decommissioned. Success will only be achieved once the old code is deleted entirely!

It's also important to remember that well-established tools change. For instance, the Java programming language has continually evolved, remaining relevant through many years and major revisions. This included incorporating elements of functional languages for more concise and error-free coding. User interface libraries are also constantly changing. Adopting modern tools and techniques can reduce the burden on your developers so that they don't spend time updating internal solutions that don't align with your core competencies.

So, to be successful, developers in your organization should embrace modern techniques when building your solutions. Teams should be capable of objectively evaluating the merits of keeping legacy code, rebuilding a solution, or using a third-party offering.

As you choose new tools or shiny objects, make sure you use discretion. Every year, we see new technology mega-trends from data stores and machine learning frameworks, to open source projects and languages. These garner considerable press and developer followings. Adopting too many new tools each year is a prescription for too much change. Many must-use trends turn out not to be as transformational as originally advertised. But some of those trends do become foundational best practices: polyglot data storage, data streaming, and the Python language for machine learning are all here to stay. Choosing the right shiny objects at the right time is another part of the art of software architecture.

Be willing to let go

It's important that your teams do not become too attached to existing components and solutions. Given the rapid evolution of the cloud, it is almost impossible to code your entire stack in-house. While a custom-built solution today might make sense, in a few years, your organization might be better served by an open source or third-party offering that evolves with the times.

Over your product's history there have likely been times where your team opted to build low-level machinery in house. And while that was the right call at that time, it is important to review the solution objectively and be willing to replace it if a better one emerges on the market. Evaluate whether home grown components can be replaced by external products that will save you developer time and allow you to focus on your own platform or applications.

To make these decisions effectively, you should look beyond upfront costs, considering long-term factors like integration capabilities, product roadmaps, workflow impacts, and the amount of time a solution will require to implement and maintain.

Okta's business model is built on the fact that building a robust and secure Identity solution for any product is difficult. A decade ago, most teams built user authentication and access management for their product internally. But most companies are not in the identity business and can't keep up with the mission-critical task of maintaining security best practices. Okta, like other API-first services—including payments and log-management—makes letting go of home grown solutions the right choice.

Principle #3: Partition your strengths

Having all of your developers work on all of your products and features is untenable. And even expecting senior developers to have context on most of your surface area can be an obstacle to your organization's success and growth.

Today, the concepts of microservices and bounded contexts are well established due to the reasons discussed here: smaller building blocks tend to encourage teams to iterate more quickly and recent trends in containerization have made this easier to manage operationally. But if, like many companies, you are starting with a monolithic application, it makes sense to focus on both microservices and 'macro'-services.

To begin this journey, your product could be divided logically into three or five areas, for instance. This can

be done by thinking logically about product divisions and migrating teams towards functional and technical boundaries. Over time, what was once strongly coupled technology choices can evolve independently with teams focusing on large but shrinking parts of your overall offering.

Shift to long-term thinking

When a product team starts out, it makes sense to focus on finding product/market fit. Initial architectural choices are important, but the long-term considerations are secondary. Once fit is achieved, however, teams need to think longer term about their infrastructure. A good rule of thumb here is to dedicate between a quarter and a third of your resources to non-feature-focused work including performance optimization, code refactoring, exploring new technologies, and porting to new infrastructure. Allocating less than that may eventually lead to that product's stagnation over time.

As they focus on career growth, many developers appreciate rotating into such projects for a designated time period. This can lead to a cross-pollination of talent across teams that can better position your company to innovate. Teams working on internal core initiatives should also learn and ask where the company—and its customers—are headed in two to three years. An organization should plan and execute three or four long-term bets each year to help make that vision a reality.

Achieving innovation through architecture

Ultimately, the best way to quantify your organization's success is by measuring the throughput, reliability, and innovation of your services over multiple years. If your organization is currently focused on innovation, but your teams are spending their time fixing problems rather than building new product areas, then there are likely problems with your architecture or your organizational structure. Technology companies need to ensure that they are building applications on a solid foundation and embracing a culture that encourages innovation.

To learn more about how Okta can help you achieve architectural excellence, get in touch.


okta