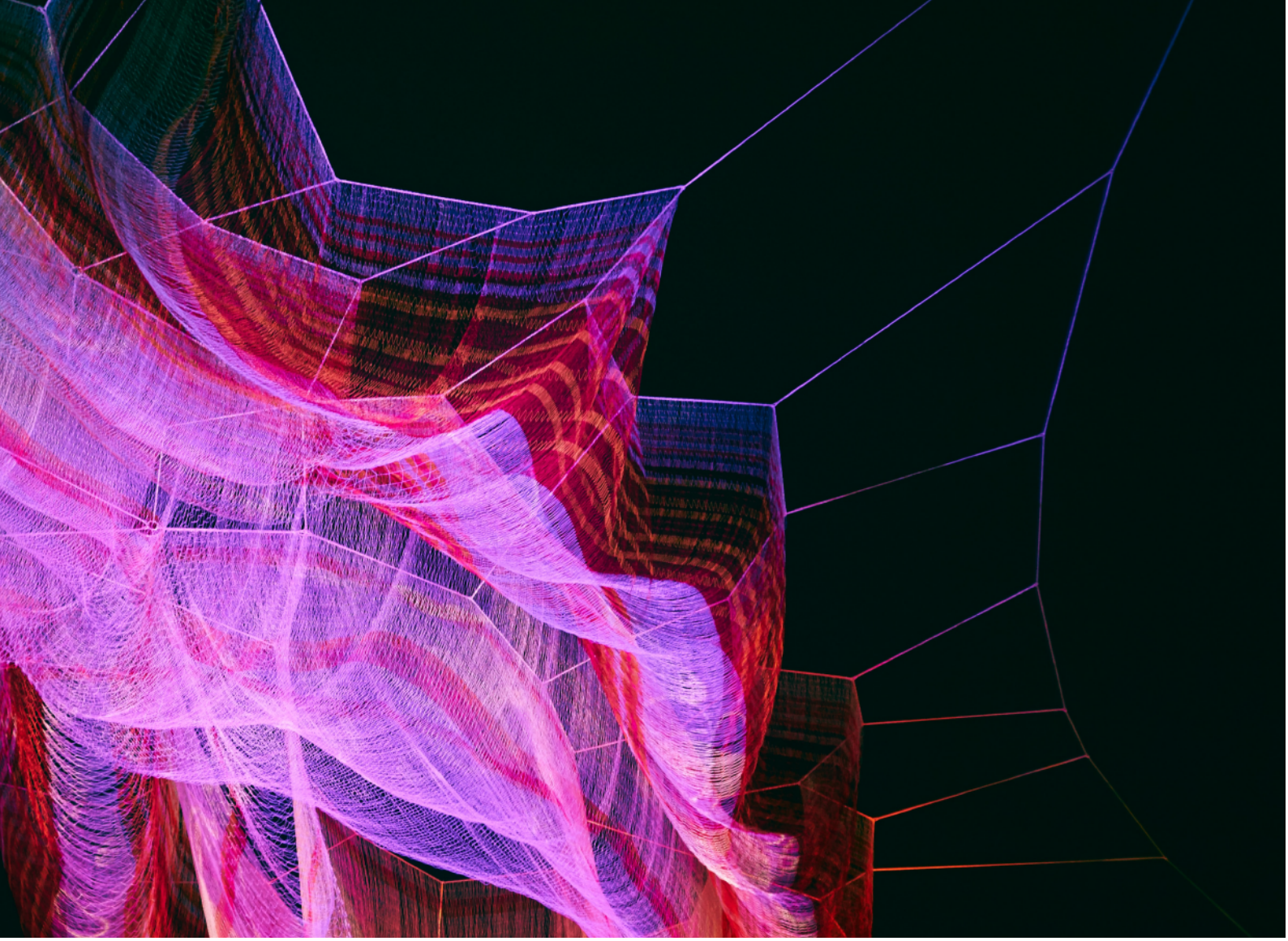


Microservices: Deconstructing the Monolith Without Breaking the Development Team Apart

" As the cloud continues to be ubiquitous across enterprises shifting toward digital transformation, new forms of virtualization are breaking down traditional siloes in application development and infrastructure operations: microservices, containers, and serverless platforms. This whitepaper demystifies microservices, containers, and serverless platforms – and the value propositions that Skygear can provide to enterprises. "

The innovation and advantages that microservices, containers, and serverless platforms bring to businesses are propelling these technologies into the mainstream, giving developers wide-ranging choices when building and deploying their projects. In fact, 63% of surveyed enterprises across the world are already using a microservices-based architecture to speed up their time to market, enrich their customer experience, and streamline their business processes. By 2023, 70% of global enterprises will be running more than two containerized applications. Going serverless is gaining traction, too: 21% of surveyed businesses are already using serverless technologies in their cloud deployments.

Microservices are an architectural pattern where various loosely coupled services work together to form an application. Each service serves a single purpose: to encapsulate all related logic and data. These services communicate with each other via well-defined APIs. To see the advantages of microservices and containerization, look no further than Netflix, Google, Amazon, eBay, and Twitter. Their transition to microservices-based architecture has become crucial in their technology canon.

Here are some of the areas that a microservices-based architecture can address:

Scalability and availability

A microservices-based architecture solves several challenges of packaging and deploying applications by modularizing them. Netflix, for example, transitioned from a monolith into a cloud-based, microservice architecture to improve the scalability and availability of its services.

At the time, Netflix could not build data centers fast enough to keep up with the exponential growth of its user base. A microservices-based architecture enabled Netflix to scale components of its applications up or down, thus saving computational resources. And given the nature of Netflix's service, outages are out of the question. Adopting a microservices architecture helped Netflix minimize the negative impact of bugs and sustain certain service levels even if a component in its applications is not working properly.

Communication overhead

A microservices-based architecture eases the cognitive load for developers. When a complex piece of software is broken down into independent subsystems, enterprises can better organize their development teams around these services. Developers need only care about the codes of the subsystems they work on and don't have to worry about how their work may affect the other parts of the software. Communication between these subsystems are done through pre-defined

messaging protocols, enabling them to communicate clearly and consistently and thus reducing communication overhead and conflicts in their own workflows.

Choice of technology stack

In a monolithic architecture, most development teams are limited to the technology stack they are using. This can discourage the team from exploring new technologies, not to mention hiring developers who use them. This is exacerbated by a cloud and digital skills shortage. In a recent Gartner study, executives cited their pace of change — transformative strategies that mitigate an organization's exposure to disruptions — the need to keep up with digitalization and emerging technologies, and talent shortage as the major risks to their businesses' growth. Microservices liberates the team from the constraints of their own technology stack. Services communicate via APIs, so it's no longer necessary to write all the services in the same programming language. In a microservices-based architecture, a team, for instance, can flexibly use .NET as the back-end technology while writing services in Python.

Microservices and containers: The best of both worlds for cloud-native app development

Containerization packages an application so that it can be run with its dependencies, isolated from other processes. When an application is shipped via containers, it is guaranteed it can be run consistently in all environments — significantly reducing errors and communication overhead caused by inconsistencies and incompatibilities. Container orchestration tools, for example, Kubernetes, further reduce the maintenance costs of containerized servers, as they automate operations — scaling the apps up or down, monitoring and repairing the containers, and balancing the load of containerized services, among others.

Adopting a microservices-based architecture and containerization technology can provide the best of both worlds. Microservices offer scalability and availability through modularity, while containers provide automation. Microservices reduce communication overhead between development teams while containers smoothen frictions between different teams (e.g., development, deployment, monitoring, auditing) — a notable boon for enterprises that are already adopting DevOps.

Microservices and containers are an indispensable part of cloud-native application development. In fact, most of the cloud providers today provide managed services for containerization technology, such as Azure Kubernetes and Google Kubernetes Engine.

Skygear: A serverless platform for mobile and web app development

A caveat to reaping the full benefits of microservices and containers is overcoming a steep learning curve. Implementing, maintaining, and securing a containerized, microservice infrastructure require technical skills and expertise that surveyed enterprises reported as major hurdles hindering them from fully adopting microservices, containers, and especially Kubernetes.

Oursky has been using Docker ever since its release. We have also deployed a company-wide Kubernetes cluster to encourage our developers to explore new technologies they can integrate it with. Our experience with Docker and Kubernetes led us into creating a serverless platform that can automate our workflows on Docker and Kubernetes while lowering the costs of onboarding other developers.

Skygear: Going beyond serverless

Most serverless solutions focus on cloud functions or microservices hosting. As a development-oriented company, Skygear provides a complete suite of application backend solutions that goes beyond hosting cloud functions. Figuratively, Skygear is like Ruby on Rails applied on a monolith, but for microservices — an optimized app platform that empowers developers with the framework and tools to build better apps faster.

Skygear is a multiplatform and serverless microservices framework designed to make application development easier and more efficient. Here are some of Skygear's value propositions:

What is serverless computing?

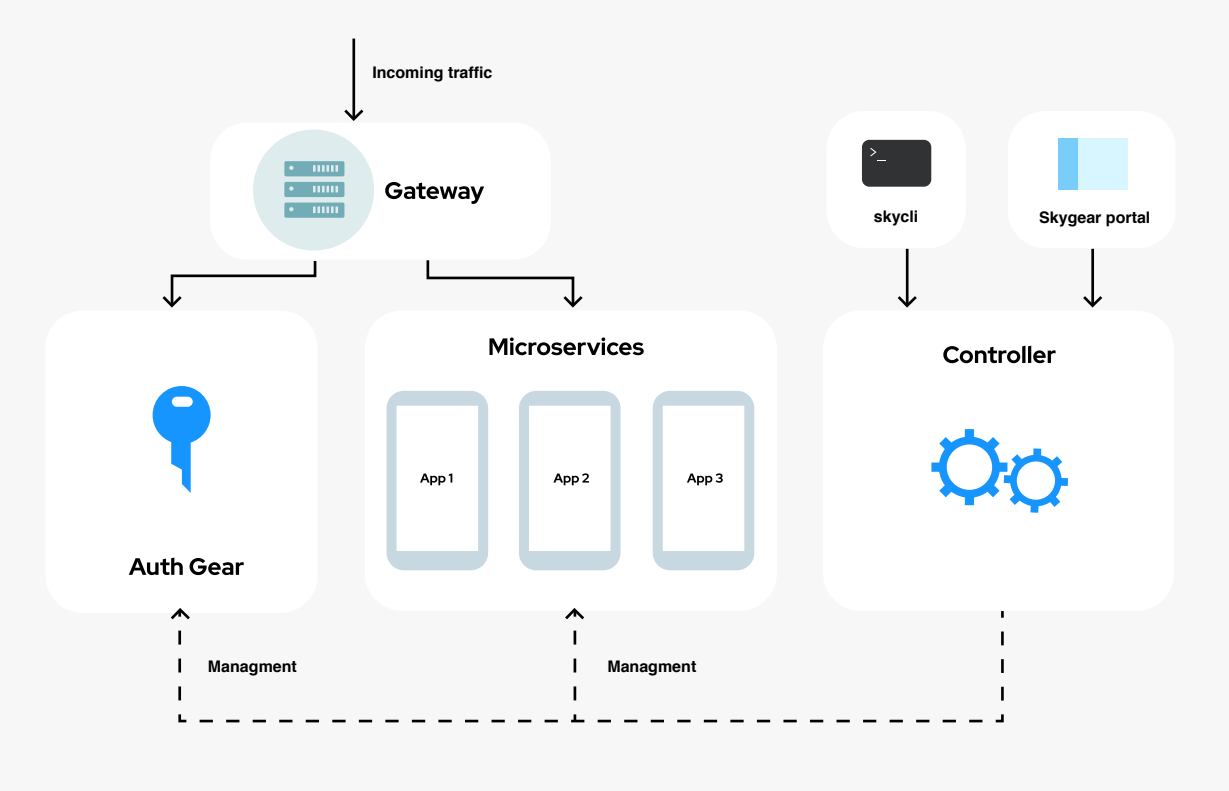
"Serverless" is a misnomer. It does not mean the absence of servers, whether physical or virtual. Serverless computing allows developers to create applications that don't rely on a dedicated server, virtual machine, or container; only the code runs on a cloud server until it completes its task. This frees developers from lock-ins, allowing them, in turn, to just focus on what matters most: writing and shipping codes.

- ✔ Modernized infrastructure. Skygear is a k8s-based serverless platform that helps accelerate transition to containers and microservices, which are fundamental to cloud-native app development.

- Agility without disruptions. Skygear clears development and timeline bottlenecks by automating the Docker and k8s pipelines, defining a microservice architecture, and providing authentication mechanisms out of the box.

- Effortless and low-cost deployment: Skygear helps optimize startup and deployment at minimal cost by freeing developers from the need to set up and maintain infrastructure.

- Collaboration-oriented adaptability. Skygear has built-in support and services for multiple platforms, enabling developers to access APIs across different platforms (i.e., Android, iOS, JS SDKs). This spurs collaboration and enhances agility among developers so they can build apps together regardless of their environments.



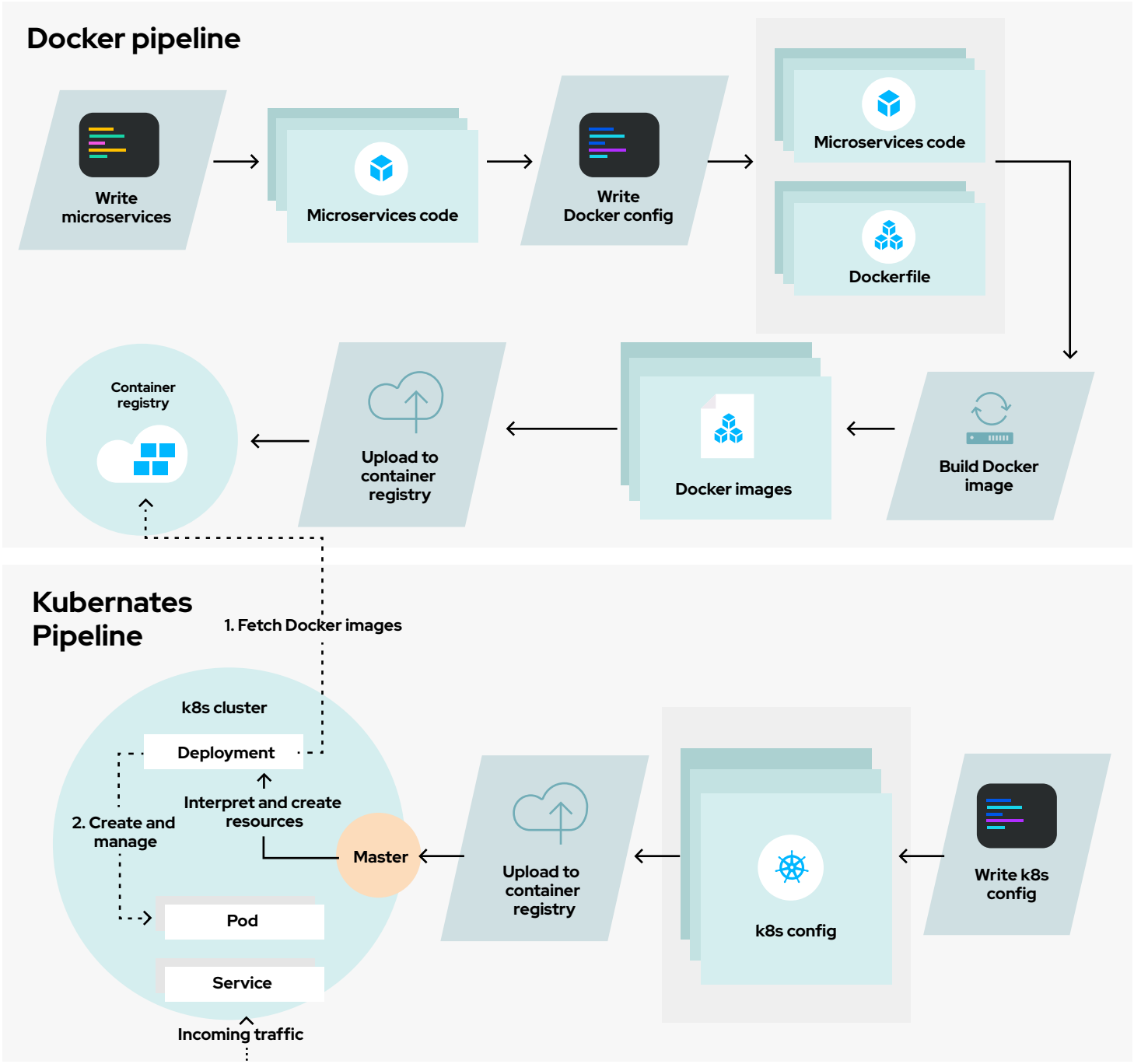
Skygear Architecture

Docker and Kubernetes automation

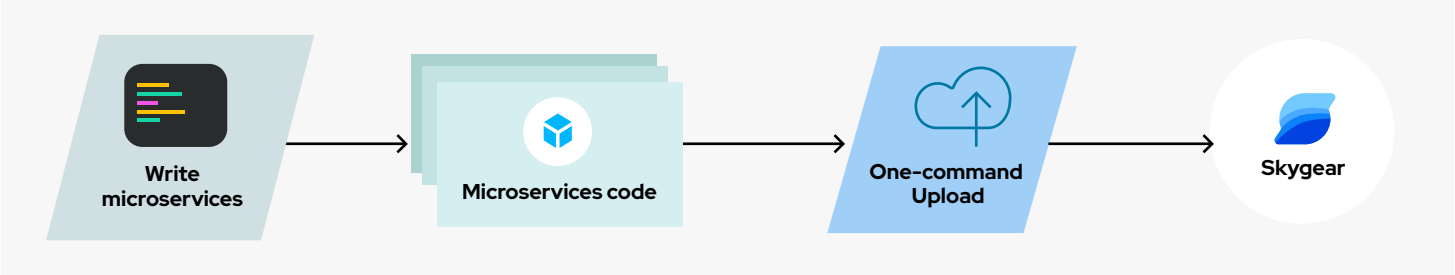
DevOps as well as IT and infrastructure teams often have to navigate various complexities in order to set up a Docker and k8s pipeline. For one, scripts need to be written to enable developers to publish Docker images to a Container Registry. A k8s cluster would then have to be set up, configured, and pointed to the right Container Registry. Additional scripts are written in order to fetch the right Docker images from the Container Registry so that developers can deploy their works in Kubernetes.

As a serverless platform that rides on container technology, Skygear helps developers through pre-configured Kubernetes clusters that already incorporate best practices and security.

Skygear also automates the pipeline of creating a Docker image and deploying it to Kubernetes. When developers ship their codes onto Skygear, a Docker image is built then deployed to Kubernetes. The whole pipeline is managed by Skygear, eliminating the constant need for their maintenance and minimizing the risk for misconfiguring the Docker and Kubernetes configuration files. Through **skycli**, Skygear's command-line tool, developers can deploy a microservice into Skygear with a single command.



How Docker and k8s pipelines are set up without Skygear



How microservices are easily set up with Skygear

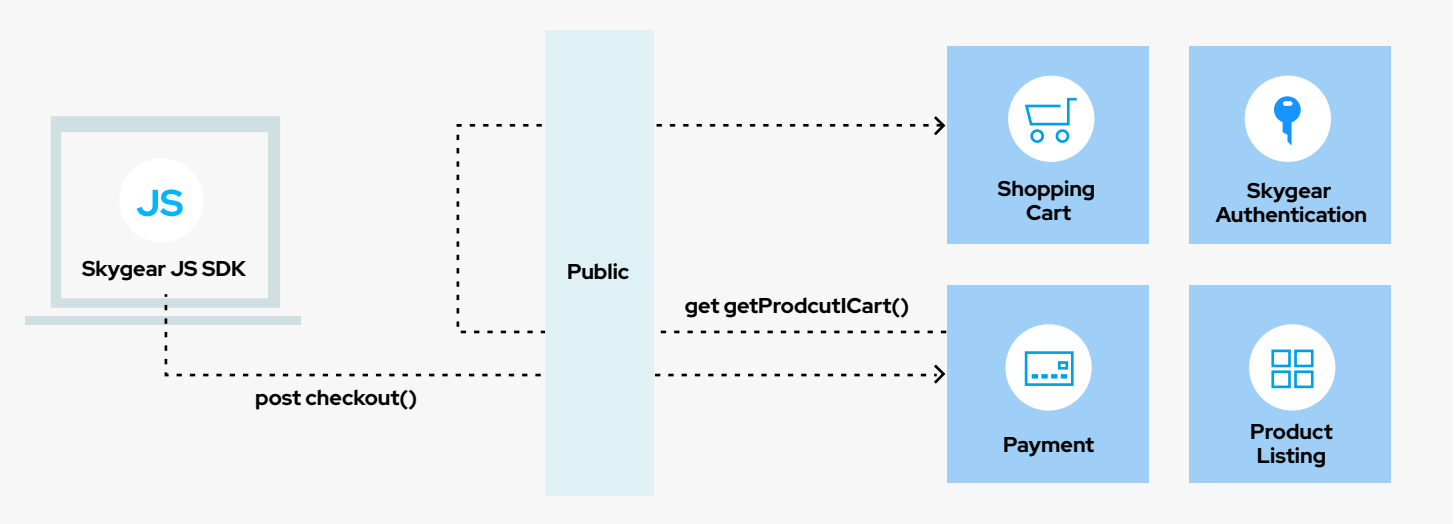
Microservices infrastructure management

A major challenge in implementing a microservices-based architecture is managing the messaging protocol and the discovery of services. There are a plethora of services and protocols that can be used by microservices to communicate, which can lead to misconfigurations if not properly set up. Skygear simplifies this by standardizing the communication between microservices through RPCs and job queues:

- Remote Procedure Calls (RPC) — Skygear recommends synchronous call between microservices via a common gateway over encrypted HTTP/2.0, so that authentication, authorization, and other security features such as rate limiting could be easily enforced for both public and microservice-specific endpoints. Treating each internal microservices like third parties is a best practice for a healthy and scalable architecture.

- Job queues — Skygear has a built-in job queue function that enables microservices to hand over or pass a job for another microservice to handle or process. This is recommended for asynchronous or background jobs.

Microservices in Skygear communicate via a public gateway in HTTP. For example, a simple e-commerce website often has three custom microservices in the application: product listing, shopping cart, and payment. These microservices are in an HTTP server written in Node.js. When a user checks out, a request will be sent to the backend via the JS SDK. A request will, in turn, be sent from the payment to the shopping cart microservice to obtain all the products in the shopping cart via the public gateway.



How Skygear works for a simple e-commerce site

Security by design

Skygear comes with an authentication microservice, Skygear Auth. It can be easily integrated into the application via the iOS, Android, and JS SDKs, helping manage the app's user sessions. Skygear Auth is tightly integrated with microservices and cloud functions deployed in Skygear to provide authentication and authorization over API endpoints.

With Skygear, agility and security are not mutually exclusive. Skygear's authentication services help accelerate the development timeline and lifecycle. As a software development company that works on over 50 projects per year, it is strenuous to re-engineer an auth services from scratch for every new project. Turning common user authentication features into a reusable microservice significantly helps save time.

Skygear's authentication services also helps developers "shift left" by incorporating security early into the development lifecycle. Many data breaches are often due to weak authentication mechanisms and improper configuration, such as storing passwords in plain text, or focusing too much time on development that security controls become an afterthought.

Essential security mechanisms in Skygear are built-in, enabling security and privacy to be baked into the development process regardless of whether developers have the resources to implement or augment them. Skygear's security features adopt the best practices and benchmarks of OWASP's Application Security Verification Standard (ASVS) and Mobile Application Security Verification Standard (MASVS), are regularly monitored through external and internal audits, and are applied with the latest patches.

Form-based authentication	Common form-based authentication, including sign-up and logins via combinations of passwords and username or email
SSO integration	JWT and OAuth 2.0; with native support in Google, Facebook, LinkedIn, Instagram, and Microsoft Active Directory
Welcome email	Out-of-the-box SMTP server; email templates can be sent in rich and plain text format via the Developer portal
Email verification	Out-of-the-box SMTP server; email templates can be sent in rich and plain text format via the Developer portal
SMS verification	Verification token via SMS; currently supports Nexmo and Twilio, and developers need to bring in their own accounts
Forgot password	Out-of-the-box SMTP server; email templates can be sent in rich and plain text format via the Developer portal

Authentication services/features currently supported in Skygear

Multiple factor authentication (MFA)	Security token and password; MFA methods currently supported are SMS, email, and time-based one-time password (TOTP)
Session and device management	Listing active sessions to detect suspicious logins and revoking them in case of compromise
Password policy	Password with specific requirements, which developers can set through the portal; currently supported are password length in characters, upper case, lower case, digit, symbol, password complexity (calculated by Skygear), excluded keywords, rejection of old passwords
Passwordless Llogin	Passwordless login methods currently supported are email and SMS
Forgot Password	Checking the password entered against existing pwned password database

Security-related authentication features currently supported in Skygear

Account and password management	Password hashing; verification and password reset sessions; passwordless, 2FA, WebAuthn, weak/pwned password checks
Gateway	Authentication and authorization; encryption in transit
Network	Rate limiting and other protection methods for distributed denial of service (DDoS)
Credentials	Secrets management
Encryption	Field-level, end-to-end

Security features currently supported in Skygear

Shared responsibility

Skygear is made for developers — by developers. Skygear embodies the model of shared responsibility — from people and process to technology — in developing user-friendly and feature-rich yet secure and compliant applications. Skygear has been open source as well as vendor- and cloud-agnostic since its release, and can be portably deployed to any cloud infrastructure, be it AWS, Azure, or Google, or on-premise systems.

Many enterprises are already harnessing the power of microservices, containers, and serverless platforms. However, they could not have done it successfully without the right tools. Poorly implemented and improperly configured microservices and containers can lead to more complexities, superfluous loads, and, in worse cases, compromises. Just as enterprises moving towards digital transformation need everyone to get on board, transitioning to microservices requires the same effort — an aspect that Skygear can help businesses with.

Skygear complements the advantages of adopting microservices, container technologies, and serverless architecture. Building, shipping, and deploying an application with Skygear can significantly reduce the operational overhead and security impact on the company — ensuring agile, flexible, and scalable yet secure software delivery cycles.

Learn more about Skygear at **https://skygear.io.**

Microservices: Deconstructing the Monolith Without Breaking the Development Team Apart

Learn more at skygear.io