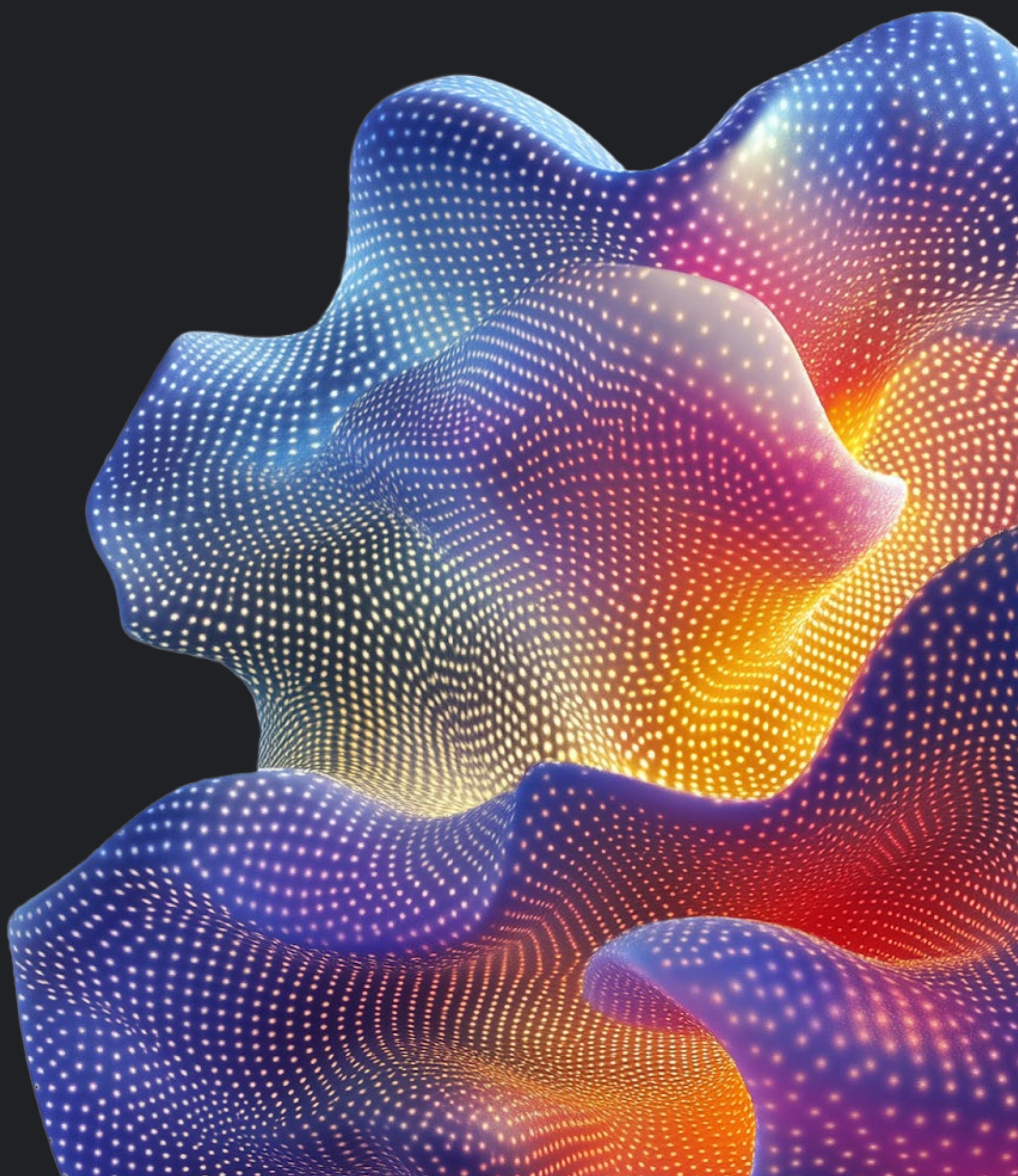


# Three pillars of a modern, AI-ready platform

How the right platform strategy  
will help your teams succeed with AI

By Richard Seroter  
Chief Evangelist,  
Google Cloud







# Table of contents

Inside this guide	03
Introduction	04
Prompting a new approach to platforms	07
Pillar 1	
The purpose of a modern platform	10
Pillar 2	
The pieces of a modern platform	20
Pillar 3	
The processes for a modern platform	28
Google Cloud is your platform partner	31
Next steps	33





# Inside this guide

Are your developers using the latest AI-ready platforms to power ahead with innovation? Or are they taking an ad-hoc, case-by-case approach to building applications and services? Discover how modern, AI-ready platforms help accelerate development and deployment, delivering tangible business benefits along the way.





# Introduction

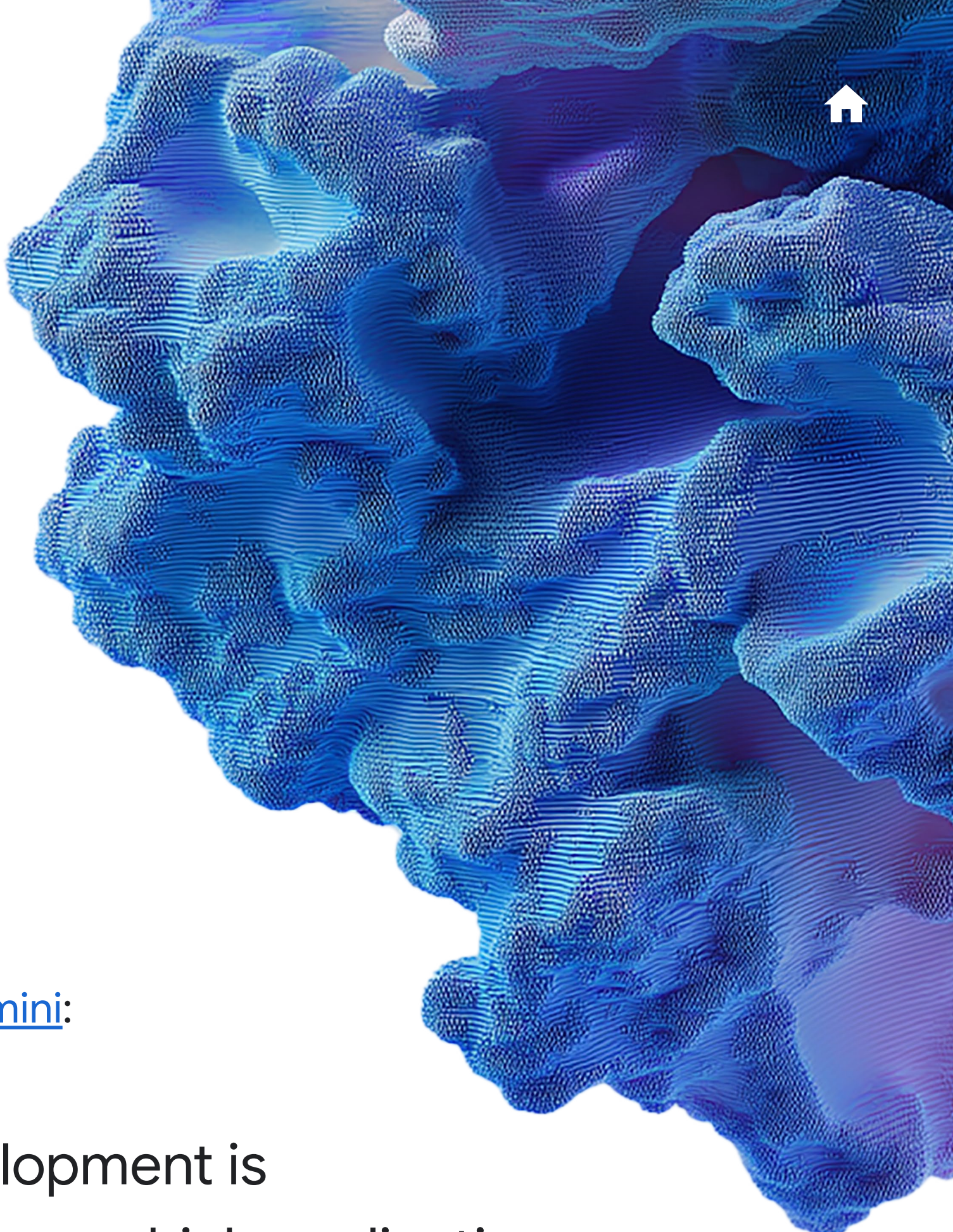
Years ago, I bought a house that didn't have air conditioning. "It's fine," a friend said. "You only need A/C for a week or so each year in the Seattle area." His advice may have been sound back then. But today, with a changing climate, it's no longer apt. Our homes play a big part in our lives and our needs shift over time—sometimes, what we started with doesn't suit our current needs.

Needless to say, I've added an A/C unit to the house.

It's a similar story for the platforms that power enterprise technology systems. As generative AI evolves and becomes production-ready, companies around the world are revisiting their existing platform strategies and undertaking important upgrades to ensure developers are fully supported to make the most of these technologies.

**In this paper, we'll explore the three pillars of a modern platform strategy: purpose, pieces, and processes of a modern, AI-ready platform strategy—one that serves your needs today and tomorrow.**





# What is a platform?

With platforms clearly powering the AI revolution, it begs the question: what is a platform? For the purposes of this paper, let's go with the definition [created by Gemini](#):

A platform in software development is essentially the foundation upon which applications or services are built. It provides a set of tools, services, and infrastructure that developers can leverage to create their software products.

Platforms help establish consistency at scale, abstract away complex infrastructure, provide centralized governance, and offer developers a self-service interface for deploying and running applications. Instead of wrangling infrastructure or reinventing the wheel, developers can spend their time building valuable services for customers.

Of course, platforms are not suitable for every use case. Some scenarios call on deep technical expertise or specialized services, in which case a decentralized approach to deploying gen AI solutions may be better. But for rapid time to value, enterprise-wide deployment, lower total cost of ownership, and scalability, platforms come out on top.

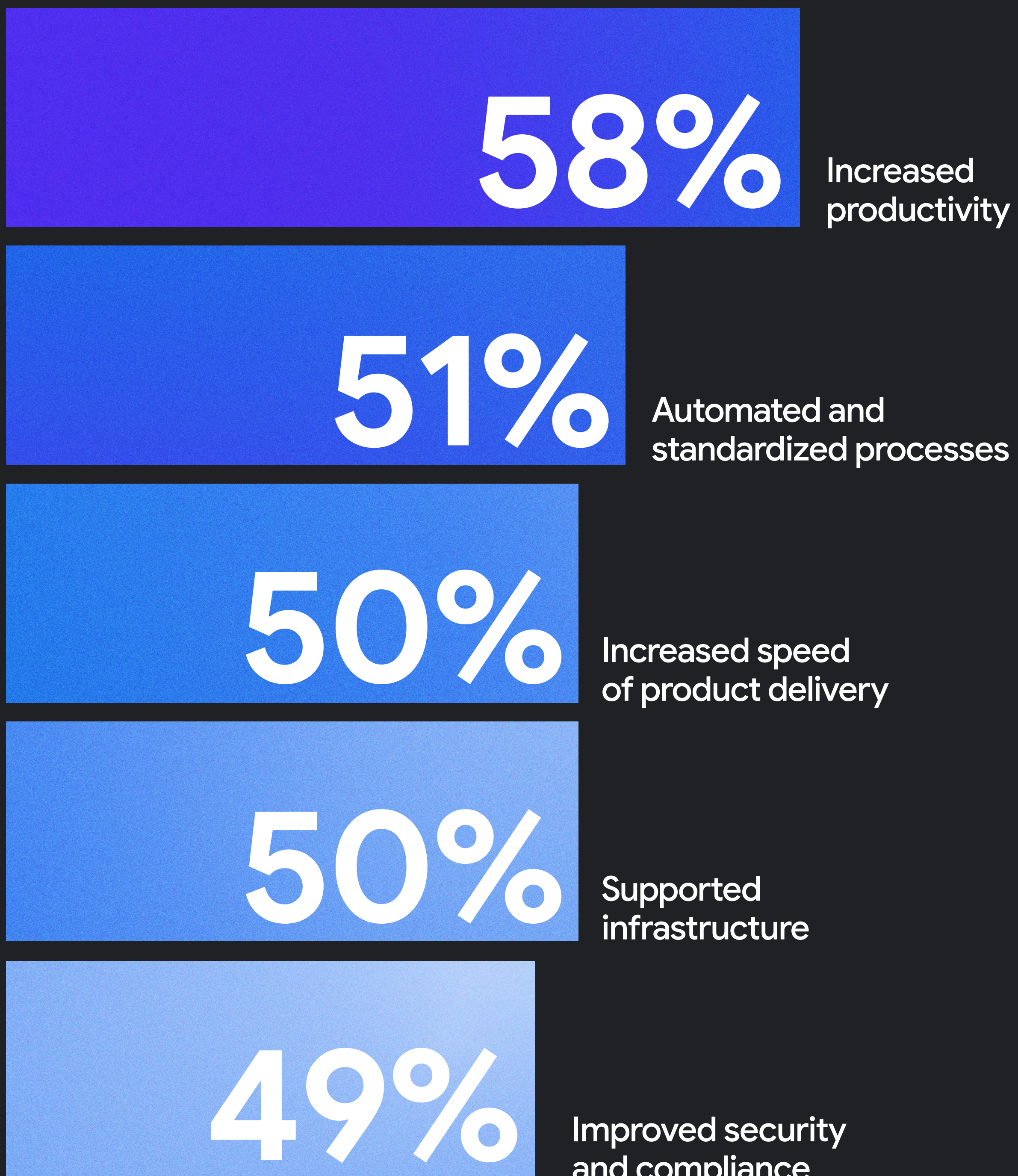
An organization will almost always have multiple platforms to serve different needs: a container platform, a developer platform, an on-premises platform, an AI platform, and so on. These platforms can be built or bought. With the proliferation of platforms, a new discipline called **platform engineering** has emerged that helps to put structure around how to approach your platform strategy.



# What is platform engineering?

Platform engineering is the act of actively designing, building, and maintaining platforms for internal customers. It's important to emphasize maintenance here—platforms are continuously evolving to meet the changing needs and constraints of various internal users.

## Top 5 goals of platform engineering<sup>1</sup>

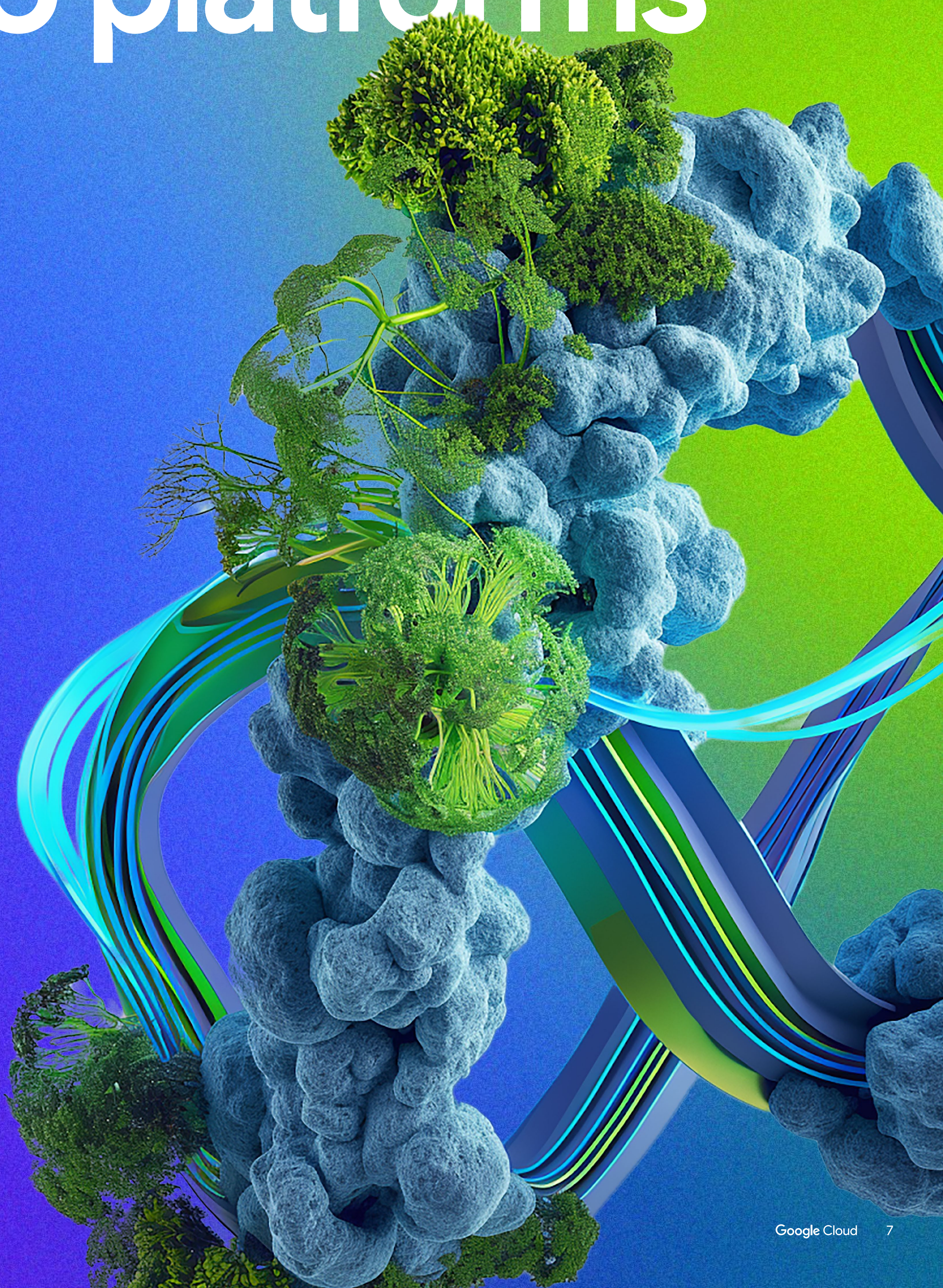


1. Puppet by Perforce, 2024, [State of DevOps Report: The Evolution of Platform Engineering](#)





# Prompting a new approach to platforms







There is no doubt that organizations are moving beyond experimentation with gen AI.

## Today, three in four organizations are already seeing ROI from their gen AI investments.<sup>2</sup>

They're reporting benefits ranging from cost savings and operational improvements, to competitive differentiation and customer service excellence. The rapid shift from pilot to production is prompting organizations to reassess their platform strategies. Business leaders want to know if their infrastructure is optimized for AI, and whether their current platforms help accelerate time to market, realize competitive advantage, and deliver the best returns. And IT leaders want to know whether their platforms are striking the right balance between reliability and the need to innovate at speed.

All these questions are important. Without the right platforms, organizations will struggle to bring AI to life—let alone accelerate adoption.

See, AI workloads introduce unique challenges and demand new conditions from the technology environment. They need inherent scalability, along with the easy ability to balance cost and performance. On top of this, AI is a complex tech stack with ever-evolving libraries and tools. Keeping pace with this constant stream of change is challenging, as is the assurance that AI-powered apps can run securely and compliantly. It is front of mind right now, with 90% of decision makers regarding optimizing workloads for AI as a top priority for the next 12 months.<sup>3</sup>



2. Google Cloud, 2024, [The ROI of generative AI](#)

3. Google Cloud, February 2024, Enterprise Jobs to Be Done Research, Internal report





Organizations will have teams that approach the problem differently, depending on their timelines, capabilities, and budgets. Some of their teams will buy fully managed AI platforms, and other teams may opt to support their own AI initiatives—building, training, and serving models on platforms they already use. To take full advantage of innovation possibilities, while limiting toil, organizations should adopt a deliberate platform engineering approach.

According to Gartner, 80% of large software engineering organizations will establish platform engineering teams as internal providers of reusable services, components, and tools for application delivery by 2026—up from 45% in 2022.<sup>4</sup>

For platforms, it's clearly time to shine. And the good news? Most likely, your organization already has a platform strategy—even if you don't call it that today.



**Even before our cloud migration, we embraced a platform approach and platform thinking. We know we can go faster and be better if we have reusable solutions.”**

Jacek Ostrowski, Senior Director of Platform Engineering, Sabre





The first pillar:

# The purpose of a modern platform







Whether you're building your first platform or your fiftieth, you need to start by asking, "Why?". After all, a new platform is another asset to maintain and operate—you need to make sure it exists for the right reasons.

To build your case, look at who the platform is for, what its goals are, and how you will measure success.



**Increasing the productivity and velocity of our product and engineering teams was paramount. We needed to be more flexible for customers and make it easier for them to search, price, and book faster.”**

Martin Brodbeck, CTO, [Priceline](#)





# Google's platform philosophy

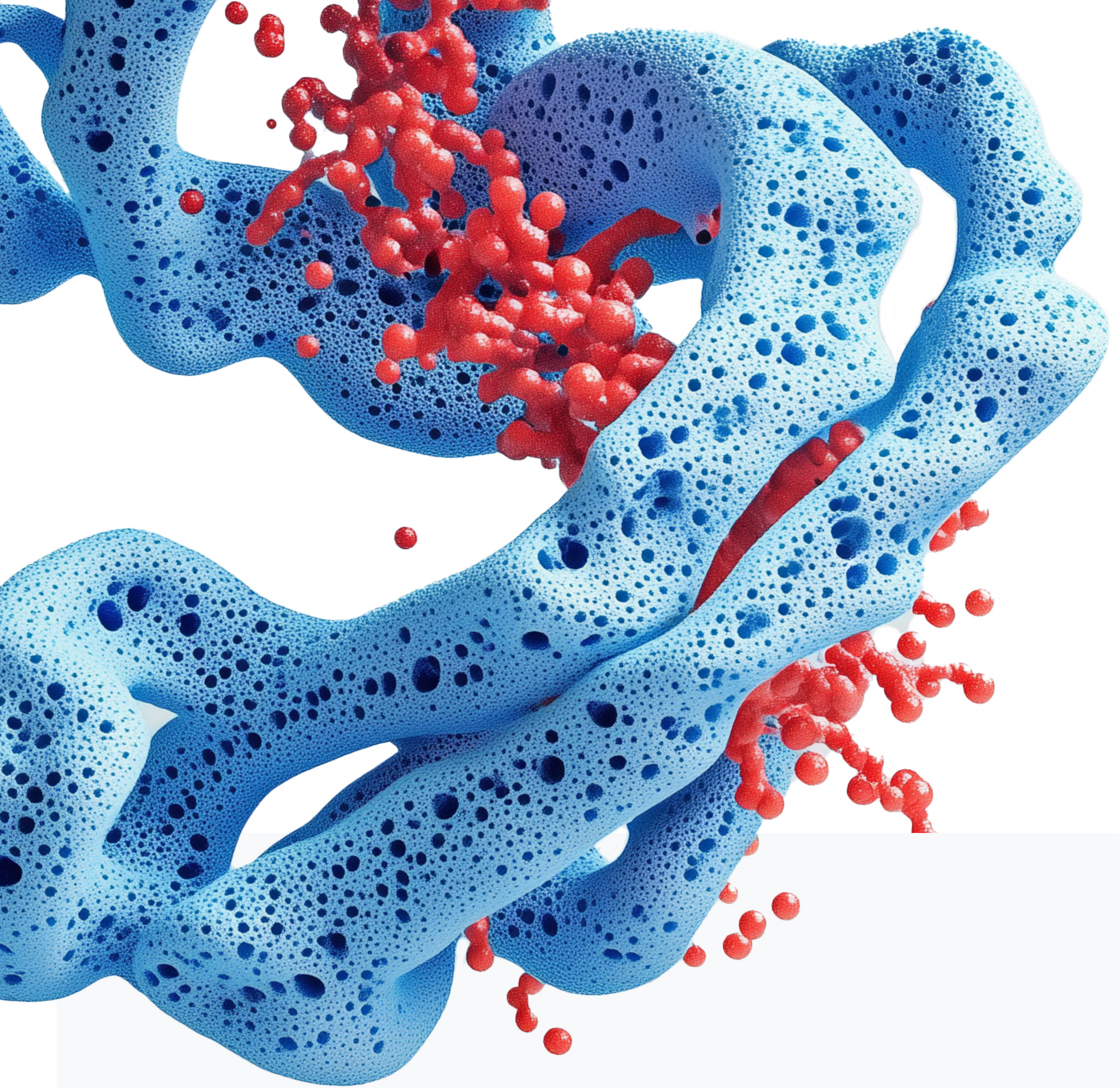
At Google, we believe strongly in the value of delivering rich platforms that improve the developer experience. We don't ask our software engineers to own the “full stack”; rather, if there's a shared technology need, we try to deliver it as part of a common platform. Lately, this means [adding gen AI features to internal platforms](#) to give our engineers the best developer experience. So, for example, by integrating AI into various stages of the software development lifecycle—including code generation, review, testing, and debugging—we've increased developer productivity, improved code quality, and optimized the development process.

**Want a deeper dive into our approach to platforms, including our perspectives on source control, build systems, change management, continuous integration, and more?**

Explore these two resources:  
[Software Engineering at Google](#)  
and [Site Reliability Engineering](#).







# What's your platform philosophy?

If you don't have a platform philosophy yet, we recommend you create one to help keep your strategy on track. The rest of this chapter should help you get started.

To learn more, read [the container platform for the next decade of AI and beyond](#).





# Who is the platform for?

Unlike traditional IT systems management, the people who use platforms are considered “customers,” not users. To deliver continual value, platform builders need to think carefully about the needs of this target audience.

A platform’s customers can include:

## Developers

These are your main customers—you don’t stand up an app platform if you don’t have app builders. A good platform serves developers with useful capabilities that make it easier to build, deploy, run, and operate software. Integrated experiences are lit up as “golden paths,” making it easier for developers to do the right thing.

## Architects

A modern architecture team is focused on keeping technical debt to a minimum while supporting a vibrant ecosystem of technologies that add business value. Architects value a solid platform built on best practices and standardized where possible.

## Product teams

Software delivery is a team sport, and product teams care about a platform that speeds up software delivery while minimizing operational toil. Platforms shouldn’t be a collection of random parts. Rather, they should consist of an intentional set of services that serve the needs of teams responsible for owning the application lifecycle.

## Site reliability engineers and Ops personnel

A quality platform makes it simpler to manage applications at scale. SREs and system administrators embrace platforms that give them robust automation capabilities, sufficient telemetry, and integration with familiar operational tools. Bonus points if you bake in generic mitigations.



## Data scientists

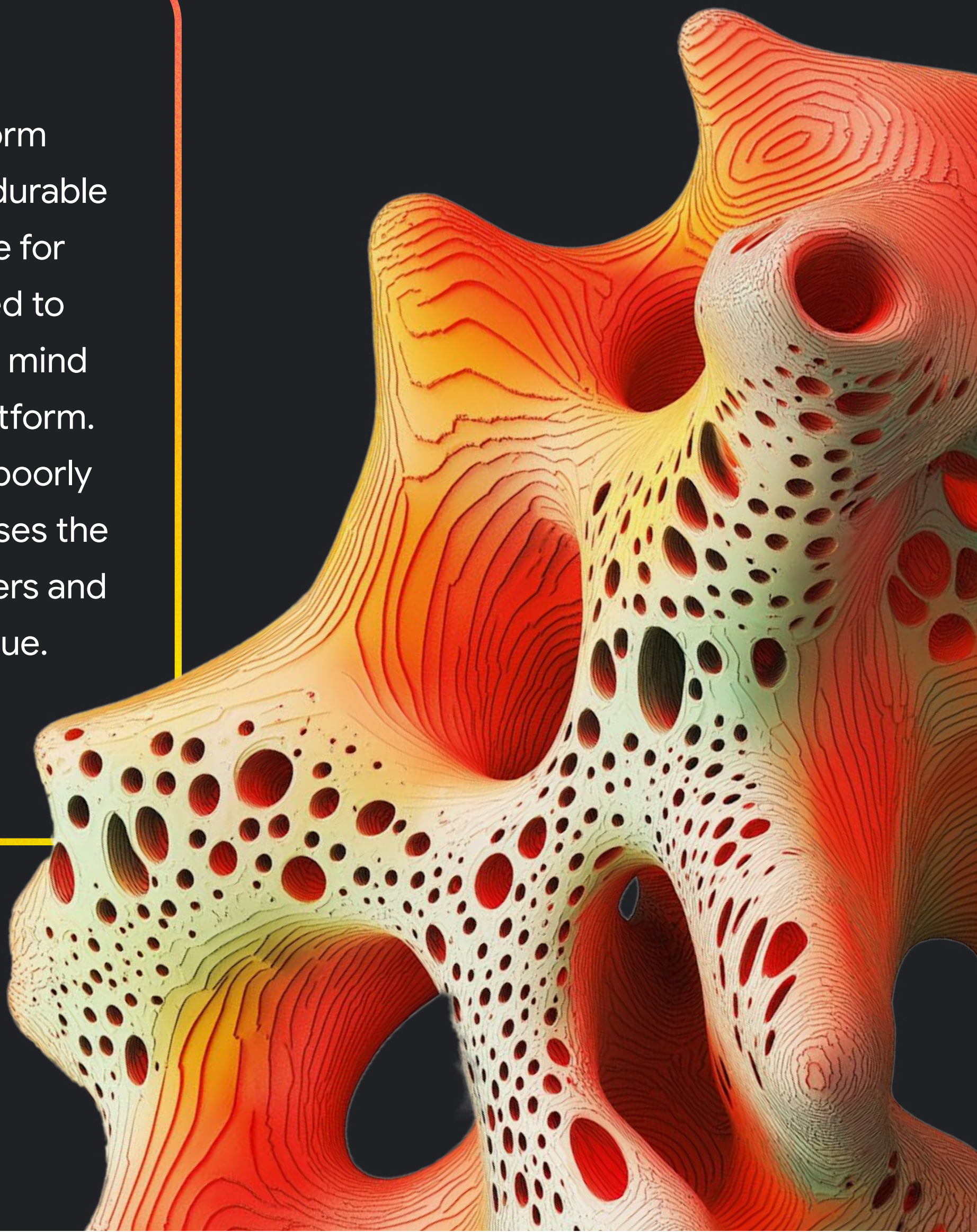
Historically, the work of data scientists lived in its own silo—far from the application platform. But with the mainstreaming of AI/ML initiatives, many of the core capabilities needed by data scientists are at home in a shared platform.

## Security teams

A well-designed platform implements security best practices across the company and improves your security posture. Everything from component patching to threat management becomes more straightforward when you've got workloads on a unified platform.

## Platform owners

When you treat your platform like a product, you have a durable team of people responsible for the platform itself. You need to keep these superheroes in mind when scoping out your platform. An unfocused, manual, or poorly integrated platform increases the cost to your platform owners and reduces your long term value. Build something that you can maintain.



**Identify your stakeholders by name if possible, and ensure your platform has the right customers lined up.**



# What are the goals?

The next step is to consider *why* you are standing up a new platform. In short, what problems are you trying to solve?

A key driver for platform modernization is to **optimize for AI**. While modern technology platforms are increasingly built with AI in mind, there are gaps to be filled in many organizations. Today's platforms need the ability to handle more demanding AI workloads and data requirements, so organizations can develop and deploy innovative new services and solutions faster. They need to be flexible enough to accommodate and integrate managed AI services alongside application services.

Another common goal at the start of the journey is to **speed up software delivery**. When each team provisions

and manages unique infrastructure and application stacks, not only are you slowing down innovation—but you end up with accidental complexity and unnecessary inefficiency all over the organization.

You might have a goal of **increasing developer productivity**. A well-crafted platform will remove friction and help developers stay in a “flow state” longer, giving them the integrated tools and guidance they need so they don't have to embark on distracting journeys through the internet.



If you're struggling to deliver scalable systems and keep them online, then the goal of your platform might be to **improve the scale and resilience** of important applications. It takes real work to design elastic and fault-tolerant systems that can handle both heavy load and component failure—and if your platform makes it easier for everyone to create such systems, it's a massive win for you and your business.

Finally, if an increasing wave of cyber attacks has you worried, then a goal of your platform should be to provide industry-leading **security controls** that don't force your teams to compromise on delivery speed. Aim to provide value-added services that apply during application development and deployment, while protecting all your production workloads.



**As you can see, the goal of a modern platform isn't just to provide some lightly automated infrastructure to IT project teams. Rather, the right platform will deliver valuable outcomes that make it safer for your teams to move fast.**





# How do you measure value and success?

You don't build and run a production-grade platform for fun. Sure, it might be fun, but it's critical to deliver something with measurable value. When an executive asks, "Why are we investing in this platform?", you want to be able to answer, "Because it delivers a meaningful ROI."



# Here are three viable ways to measure the value of your platform:

01

## Product metrics

Convey the *business* value delivered by the platform. Have you shipped new (valuable) features at a faster rate than last year? Are you experiencing fewer reliability-related issues, thus materially lowering your support costs? Have you seen a major reduction in quality issues and reversed your sinking customer satisfaction scores? Consider a handful of attributable business impact metrics that clearly show why this platform matters and how it helps you do more with less.

02

## Stakeholder metrics

Capture data—through instrumented systems and team surveys—that shows the benefits realized by the platform's customers. Are developers reporting improved productivity and a greater ability to get work done? Do architects see a higher compliance rate for recommended data strategies? Are product teams reporting lower application support costs and a faster time to resolution when issues arise? Does your CISO keep an eye on vulnerability data and notice a quicker patching cycle that reduces company risk? The platform cannot be considered successful if stakeholders are dissatisfied with it.

03

## Platform metrics

The health of the platform itself is critical for long-term success. Is the platform team regularly meeting with stakeholders to get feedback and fresh requirements? Does the platform maintain the committed uptime SLA and patch any security vulnerabilities in a timely fashion? Is usage of the platform itself increasing, measured by the number of teams and the amount each team is using it?

Think comprehensively about how you measure success, and look to frameworks like [DORA](#) for help identifying further [measurements that matter](#).





The second pillar:

# The pieces of a modern platform





**Having done your homework, you should now be clear on the customers, goals, and performance metrics of the platform you're about to build—which means you're ready to roll up your sleeves and get started.**

Let's take a look at the components and capabilities required of a modern, AI-ready platform. As you'll see below, the capabilities needed by AI app builders and data scientists can now fit into the standard application platforms your other developers need, too. Note that some of these architectural components are satisfied by fully managed services with no operational overhead—think [Google Cloud Vertex AI](#) or databases like [Cloud Spanner](#). And other components come from open source or commercial software running in a managed platform like [GKE](#). All of these technologies come together to form your platform.



# Why integrate Vertex AI into your AI platform?

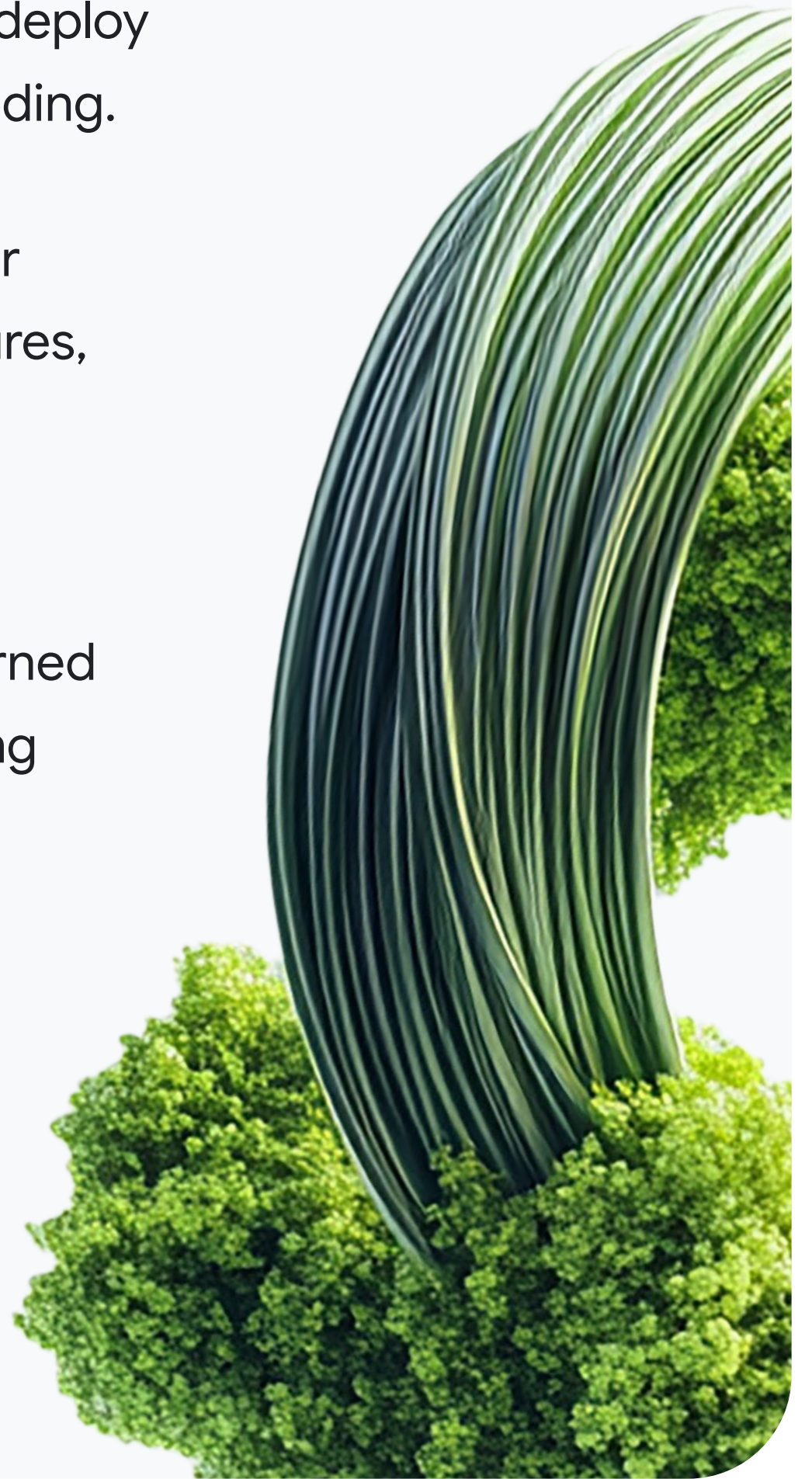
While building your own AI platform gives you a solid foundation and lots of control, integrating a managed service like Vertex AI acts as a productivity booster for experimenting, training, and deployment with advanced features that would be a real challenge to build and maintain on your own. For example:

**Agent Builder** simplifies the creation of conversational AI interfaces, allowing developers to design and deploy chatbots and virtual agents without extensive coding.

**Feature Store** acts as a centralized repository for organizing and managing machine learning features, for consistency and reusability across different models and teams.

**Model Registry** provides a structured and governed environment for storing, versioning, and deploying machine learning models, so only approved and validated models make it into production.

**Colab Enterprise** gives you a managed and secure environment for collaborative data exploration and model development at scale.



By making Vertex AI part of your platform strategy, you get access to these types of cutting-edge AI capabilities without the hassle of building them from scratch, speeding up your AI journey and opening up new opportunities for innovation.



Just remember, when building a new platform, it's tempting—indeed, easy—to over-engineer it by immediately introducing every component you can think of. Don't do that. Embrace the [thinnest-viable-platform approach](#) and keep those initial goals for the platform firmly within your sights.

That said, a modern platform will typically include the following capabilities.

## Application Platform

AI specific technologies

### Platform interfaces

api

dev portal

laC

IDE

Cloud Workstations

Jupyter notebook

### Delivery tools

source control

artifact repository

curated oss

ci/cd

code scanning

prompt repository

model registry

### App services

api gateway

service mesh

workflow engine

app registry

object storage

### Data management

database

cache

messaging

event processing

etl

vector database

### Hosting

vm

container

functions

model serving

### Security

secrets

idm

key management

### Feature management

data transformation

feature store

lifecycle management

### Observability

logging

monitoring

dashboards

model monitoring

### Model training

pipelines

parameter tuning

Platform infrastructure

Fleet operations

Knowledge repo





# Let's break down each category and their key considerations.

Category	Capabilities	Technology options
<b>Platform interfaces</b>	<p>Interact with the platform through self-service portal, IDEs, APIs, and infrastructure-as-code (IaC) or GitOps tools. Serve up cloud-based developer environments for simpler onboarding.</p> <p>To make it AI-ready, include <b>notebooks</b> and a <b>prompt repository</b>. Data scientists want a Jupyter notebook environment for training and interacting with models. Developers, data scientists, and AI app builders increasingly rely on AI prompts and will need a place to store them.</p>	<ul style="list-style-type: none"><li>• Look to support IaC products like Terraform and Pulumi</li><li>• Many teams use Backstage as their developer portal</li><li>• Consider Google Cloud Workstations for online dev environments</li></ul>
<b>Delivery tools</b>	<p>Build, package, store, and ship software to production using delivery services.</p> <p>Use a <b>Model Registry</b> to store and access models used by applications. The platform's existing container registry may also serve part of that purpose.</p>	<ul style="list-style-type: none"><li>• Cloud-based services like Google Cloud Build, Cloud Deploy, and Artifact Registry support your integrated platform plans</li><li>• Look to vendors like GitLab, JFrog, and Snyk to build out your delivery stack</li></ul>
<b>App services</b>	<p>Create low-code apps or integrations with stateful workloads, store data in object storage, and implement rich service to service communication patterns with API gateways and a service mesh.</p>	<ul style="list-style-type: none"><li>• Support app builders with products like Google Cloud Workflows, App Hub, Cloud Service Mesh, and Apigee</li></ul>





## Data management

Use relational and NoSQL databases to store app data, and introduce a cache for better performance and resilience. Set up real-time or batch integrations between systems or data stores.

Many AI apps use a retrieval augmented generation (RAG) pattern to improve the contextual relevance of LLM responses. That means having a **vector database** available in your database suite.

- Give developers a choice of relational databases like PostgreSQL or MySQL, or a cloud-native option like Cloud Spanner
- Use non-relational databases like Firestore or Redis
- Provide real-time messaging like Google Cloud Pub/Sub or Apache Kafka, along with batch processing services like Cloud Data Fusion

---

## Hosting

Run workloads in containers or virtual machines. Use a full container orchestrator or a serverless environment.

Offer a **model serving** option that's backed by GPUs or TPUs.

- Serve up the highest abstractions possible that help developers focus on apps, not infrastructure
- Use serverless runtimes like Cloud Run, and complete orchestration platforms like Google Kubernetes Engine

---

## Security services

Improve application and runtime security with secret and key management, and an identity management solution.

- Ensure that your platform offers some way to securely store confidential metadata such as Google Cloud Secrets Manager and integrate with identity solutions like Google Cloud IAM

---

## Feature management

**Features** are the measurable attributes that data scientists use to train and evaluate machine learning models. Apply engineering to transform raw data into meaningful features, store features for reusability, and provide lifecycle management such as versioning and monitoring.

- Look for software or services that provide the ability to process data, such as the Google Cloud Feature Transform Engine
- Consider services like the Vertex AI Feature Store to maintain and serve features to data scientists





## Model training

Train your models based on provided data; and run experiments and evaluate results, with options to perform hyperparameter optimization.

- Use frameworks like JAX and PyTorch while leveraging open source platforms like Ray and Apache Spark
- Explore managed services in Vertex AI for pipelines, training, and tuning

---

## Observability

Collect telemetry and analyze it to troubleshoot issues, support audit scenarios, and inform product improvements.

Offer **model monitoring** to help watch for prediction drift and any data issues.

- Take advantage of cloud-based stacks like Cloud Logging and Cloud Monitoring and frameworks like Open Telemetry
- Introduce other powerful observability services from providers like Datadog and Honeycomb

---

## Platform infrastructure

Support the overall platform with resilient compute infrastructure, high performance storage, and rich networking services.

- A platform depends on resilient underlying infrastructure such as Google Cloud Compute Engine or Google Kubernetes Engine (GKE)
- Check your processors to ensure you have the GPUs or TPUs necessary to run training jobs

---

## Fleet operations

Manage platform infrastructure at scale using engines for policy and configuration synchronization, rollout sequencing, and observability of the platform itself.

- GKE Enterprise provides Policy Controller, Config Sync, Security Posture Dashboard, and other fleet features
- Use IaC and GitOps tools to complement these built-in features

---

## Knowledge repo

Provide docs, “golden path” blueprints, fine-tuned chatbots, and other experiences that help users get the most out of the platform.

- Buy or build a knowledge repo, service catalog, failover plans, and incident response docs
- Consider products like Gemini Code Assist and Gemini Cloud Assist for chat-based interactions





# About GKE Enterprise

Many companies use Kubernetes as the foundation for their application platform. Its flexibility, rich ecosystem, and dependability make it a viable place to run many of the components listed above.

GKE Enterprise is the premium edition of GKE. The managed container platform creates a consistent platform out of a distributed set of clusters, enabling you to build and manage enterprise-grade applications at scale. With services like Policy Controller and Config Sync, you can ensure the platform stays compliant with security policies and looks identical from cluster to cluster.

It is a good host for many of the components listed above, including your developer portal, databases, CI/CD, event processor, service mesh, app hosting (with GPUs), AI/ML model training and more.

According to Forrester, organizations that use GKE Enterprise report streamlined operational efficiency, improved application availability, reduced developer toil, and greater deployment velocity—which all contribute to repurposing of legacy platform costs toward modernization initiatives.<sup>5</sup>

Learn more about [GKE Enterprise](#).

5. Forrester, 2022, [The Total Economic Impact™ Of GKE Enterprise](#)





The third pillar:

# The pieces of a modern platform







# The final piece of the platform puzzle is all about maintenance.

Why? A platform is never “done”; it’s just released. So, if you’re going to invest in a platform, you’ll need to adopt a **continuous improvement** mindset and establish a platform team to not only keep the underlying stack patched and secure, but to continually find new ways to introduce value to stakeholders.

## With this in mind, what processes should you consider to keep your platform healthy and relevant?

First, treat your **platform like a product**. Stay true to the platform’s purpose, keeping a durable team in place to build and run the platform. This includes an assigned product manager who builds relationships with developers and maintains a backlog of features and fixes to apply to the platform. They can also promote the platform to internal stakeholders to grow the user base and maintain product-market fit.

**Platform engineering** builds on the idea of running your platform like a product, but also looks at how you apply an engineering mindset to the solution. This means you’re intentionally building an integrated stack, instead of offering a pile of disconnected infrastructure parts. It’s about creating automation and APIs for platform owners and platform users—making it faster, cheaper, and easier to deliver software.

This relates to the topic of [Site Reliability Engineering](#), a concept pioneered at Google. When you treat operations as a software problem, you find yourself taking a proactive approach to latency, performance, change management, incident response, and more.





One reason folks embrace a platform approach is because they crave the **cloud operating model** of self-service, on-demand, and metered infrastructure. A platform should emit telemetry about usage and offer transparent chargebacks or showbacks to product teams, which supports a [FinOps](#) model, whereby teams can engineer and optimize solutions with cost in mind.

Fortunately, a modern platform can now rely on **AI-driven optimization processes**. Providers like Google Cloud offer generative AI-based recommendations for platform tuning. Include these recommendations in your ongoing health and maintenance of the platform to remove unused capacity, reduce your attack surface, lower costs, and improve resilience. And start exploring the rich, emerging world of AIOps which focuses on how AI can help you optimize operations.



**If you start with the user, clearly defining the problem and what you want the platform to become, you'll create something users value. Then you reach critical mass and it becomes easier and easier. A product mindset, sensitive to the needs of users, is key."**

Jacek Ostrowski, Senior Director of Platform Engineering, Sabre



# Google Cloud is your platform partner.



With gen AI adoption accelerating, many organizations are grappling with how to deliver on today's needs while preparing for tomorrow.

**The good news is  
you don't have to  
do it alone.**

Google Cloud makes the complex simple, helping you build and maintain the platform your developers need to accelerate application development and deliver your competitive advantage.

Our best-in-class infrastructure is geared for the AI era—helping you train cutting edge AI models and achieve unprecedented improvements in efficiency.

By pairing this AI-optimized infrastructure with developer, data, security, and collaboration tools built for today and tomorrow—along with professional services expertise and a culture of innovation—we empower you to become a digital and AI leader.





**Sabre and Google Cloud both have a strong engineering culture. We don't run away from complex problems—we embrace them. There's a fundamental compatibility on a cultural level. Transforming into a cloud-native company was a huge change and challenge, and the partnership helped us be successful.”**

Jacek Ostrowski, Senior Director of Platform Engineering, Sabre



**I view Google Cloud as an extension of our engineering team. The less time we have to manage and maintain data infrastructure, the more we can focus on core capabilities. Having these highly scalable, SaaS-based data platforms means we can focus on our customers.”**

Martin Brodbeck, CTO, [Priceline](#)





# When is the right time to embark on this platform journey?

There are some common triggers that prompt organizations to re-evaluate their platform strategy. These can include:

## Growth plans

Is significant growth forecast over the next 12 months? If your platform is already stressed, a more scalable strategy is needed.

## Secure AI adoption

Planning to add AI into your application or services? You'll need an AI-ready platform.

## Cost optimization

Whether it's to address financial challenges by exploring cost-saving alternatives, or to avoid unexpected cost overruns or spikes in current cloud spending, cost optimization is a key driver of platform transformation.

## Performance

Are outages and slow application response times holding you back? Security not up to scratch? Critical platform components reaching end of life? Optimizing the speed, security, and performance of your platform can deliver significant gains.

## User experience

If you're losing top IT talent due to frustrations with legacy systems, upgrading your platforms may help them do their jobs better—and keep them in-house.

## Compliance

Often, changing regulatory requirements mean you need more robust security and compliance offerings.

If one or more of these scenarios sounds familiar, then it's time to get started on your platform journey. Our Google Cloud experts can help you take your platforms to the next level.

[Book a consultation today →](#)



