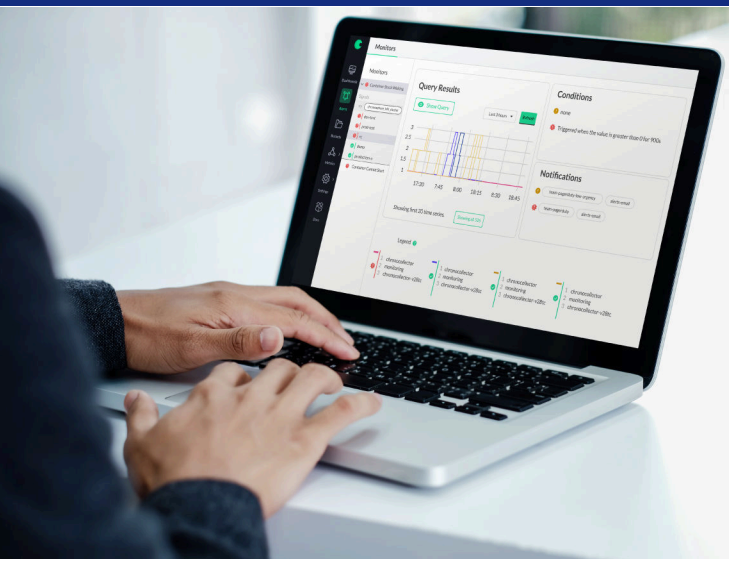
chronosphere

A Buyer's Guide to Modern Observability for Cloud Native

The keys to cost optimization and resource efficiency

Table of Contents



Maximizing the value of your technology investments with cloud native observability

During uncertain economic times, organizations will have a heightened focus on cost optimization and resource efficiency. The natural instinct is to rein in spending — often at the expense of investing in technology that makes workers productive and also attracts and keeps customers. This is a false choice. A more strategic response is to invest in tools like cloud native observability that maximize the return on your cloud native technology investments and let you keep innovating.

Whether you're a traditional enterprise modernizing and innovating for competitive advantage, or a born-in-the-cloud organization looking to disrupt the status quo, you need to get the most out of your investments in modern cloud native infrastructure and applications. And a crucial success factor in optimizing those investments is cloud native observability. Simply put, cloud native initiatives fail without cloud native observability.

A cloud native technology stack and best practices delivers:

- **A highly available and more reliable service.** Cloud native best practices enable you to build a more resilient product and service.
- **More flexibility and interoperability.** Cloud native environments are not only more portable, but they also provide the ability to scale up and down dynamically and on demand.
- **Speed and more efficiency.** Engineers can iterate faster to handle increased customer expectations. According to a Digital Enterprise Journal (DEJ) study, companies that adopt Kubernetes see a 6.8x improvement in time to market.

If not managed well, the benefits of cloud native iniatives become elusive and problems quickly surface. The shift from a traditional environment of dozens of monolithic applications on hundreds of virtual machines (VM) to a cloud native architecture, featuring thousands of microservices distributed across tens of thousands of containers, presents a unique set of management challenges:

Unprecedented data volume

Each container emits the same volume of telemetry data as a VM. Scaling containers into the thousands and collecting more and more complex data (eg: higher data cardinality) results in data volume becoming unmanageable. The unprecedented growth in observability data makes dashboards and queries run slow or not at all and makes engineers spend an inordinate amount of time just looking for the right data to fix a problem.

Unpredictable and rapidly increasing costs

The explosive growth in data volume and the need for engineers to collect an ever-increasing breadth of data has broken the economics and value of existing infrastructure and application monitoring and tools. Costs can unexpectedly spike from a single developer rolling out new code. Observability data costs can exceed the cost of the underlying infrastructure.

Complexity and ephemerality hurts engineer productivity and time to remediation

A microservices architecture features thousands of interdependent services, which makes isolating issues much more difficult and time-consuming. And some containers may only live for a few seconds or minutes, placing a new burden on engineers who are responsible for performance and reliability.

The challenge of existing tools: negative cost-to-value ratio

You want an observability solution that delivers value today and serves as a foundation for your cloud native journey — the top priority being a world-class product/service experience. That means adding capabilities today that allow your teams to locate, diagnose, and remediate a problem fast. As you evaluate observability solutions, it's important to understand why existing application performance monitoring (APM) tools, which were designed for the cloud era, don't translate well to the new cloud native world.

Observability costs exceed budget	As data grows, it's difficult to stay on top of ingestion loads and there is little to no transparency into spend.
Scalability restrictions	Rapidly rising costs force teams to restrict custom metrics and cardinality tags, negatively impacting metrics stack behavior visualization and causing teams to "fly blind."
Operational burdens	Engineers are spending long nights and weekends troubleshooting.
Downtime and data loss	Service-level agreements (SLAs) and service level objectives (SLOs) aren't being met. Small changes lead to data loss.
Lack of control	APM solutions lack data controls and visibility into observability data usage across teams and individuals. Simple code changes or new deployments can result in surprise overages.
Poor user experience	Solutions are not built for what forward-looking engineers need and want. As data grows in volume and complexity, troubleshooting takes longer, dashboards and queries are slow (or don't load at all), and engineering talent is burned out.
Vendor lock-in	Proprietary solutions make it nearly impossible to switch tools, leaving you powerless when prices go up.

The end result is that the **cost to value ratio of APM tools becomes inverted** — you're spending more and collecting more data, yet receiving less value. Continuing to use your current APM tool for containers and microservices may seem the easier path, but the decision to stand pat will soon cause problems. There will be acute pain as your cloud native environment grows. You don't want an outage that affects customers to be your compelling event to modernize your monitoring and observability. Then it's too late.

Choosing the right cloud native observability solution

Since cloud native is so crucial to efficiency and competitiveness, it's never been more important to maximize your cloud native investment with great observability. In fact,

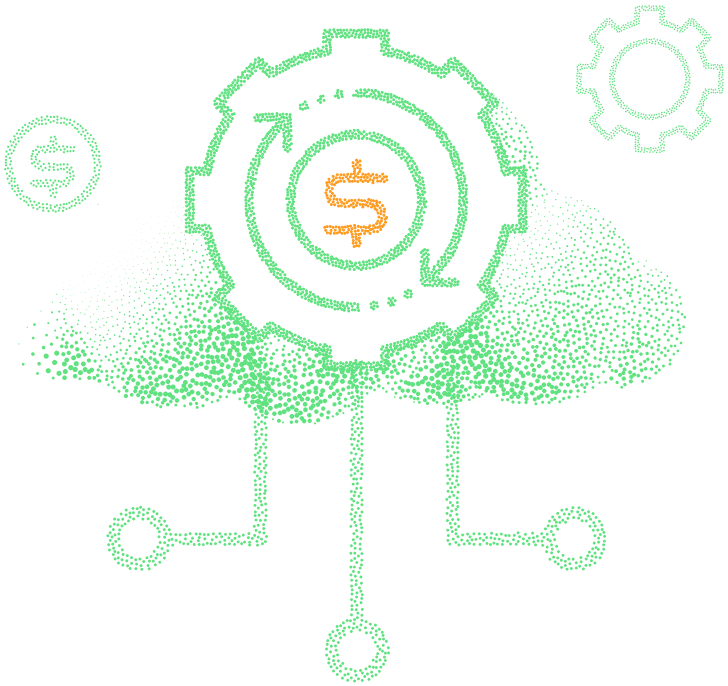
71% **of engineers say their business can't innovate effectively without good observability.**

4 key considerations

The right cloud native observability tool will give your organization transparency into costs. It will empower your teams to triage and get to root causes quickly — even in the most complex tangle of microservices. Best of all, having the data and context needed to crush the next on-call shift makes engineers happier and more productive.

1. **Tame data growth and control costs**
2. **Gain consistent availability and reliability**
3. **Increase engineer productivity**
4. **Deliver world-class customer success services**

Keep reading to discover the key capabilities purpose-built cloud native observability solutions deliver for your organization.



Tame data growth and control costs

Cloud native technologies and practices, unlike traditional monolithic IT architectures, open a whole new world of opportunities to businesses. They're comprised of independent services that communicate via APIs — not simply a large, single code base that's infrequently updated — and each service is managed by a dedicated development team responsible for frequently shipping small updates and conducting experiments to explore new possibilities.

Yet cloud native agility has also introduced increased complexity that limits visibility and creates an explosion of data that can slow down teams while they're trying to solve customer-facing problems.

As mentioned earlier, a significant way in which cloud native environments — especially as you start scaling them — differ from traditional cloud environments is that they emit massive amounts of observability data — somewhere between 10x and 100x more than traditional VM-based environments. This data alone can be challenging to manage, but the problem is made worse by developers adding more labels to their metrics, causing the number of elements in a set to spike. Existing APM solutions weren't designed to handle ever-increasing data volumes and massive cardinality, so **costs can quickly become uncontrollable** and can impact system uptime.

What's needed is a cloud native observability solution with key capabilities that helps you keep costs under control and more. Your cloud native observability solution should also help you understand how much data you have and where it's coming from as well as make it simpler to quickly find the data you need to solve issues and achieve business results.

Control Plane

Traditional APM tools lack the ability to efficiently manage the exponential growth of observability data and the technical and organizational complexities of cloud native environments. APM tools require you to collect, store, and pay for all your data regardless of the value you get from it, or, alternatively, to drop or filter data to keep costs down.

With a central control plane, you optimize costs as data grows without any surprise budget overruns. You get persistent system reliability and improve your user experience. You enjoy platform-generated recommendations that optimize performance. You also don't waste valuable engineering resources on troubleshooting, but rather reduce noise and redirect your engineers to solve problems faster. As a capability of a modern cloud native observability solution, a central control plane allows you to shape, transform, and manage your metric data based on need, context, and utility. That way you can analyze and **understand the value of your observability** data faster, including what's useful and what's waste.

Visibility and quotas

Complete visibility into how much each team or user is consuming in the observability system is **key to controlling rising costs** over time. Real-time visibility into usage can allow you to do showback or chargeback of your observability costs. You ideally want to be able to break data down into logical units that make sense — for example, by service, cluster, environment, or team — and then delegate responsibility to service owners to determine how they will use their allocated capacity. This is important as you grow cloud native beyond pockets in your organization and deploy more broadly. You want your teams to be able to easily monitor and manage their own observability data while encouraging experimentation.

Spending constant cycles on metrics reduction projects and chasing after heavy users is not the best use of an engineering team's time. The best cloud native observability solutions use protected quotas to solve the problem of one team's query or data usage crowding out another team. When a spike occurs, the system makes it easy to quickly identify the team causing it, know it's contained so that other teams aren't impacted, and start working directly with the implicated team to address it.

Downsampling and aggregation

Both downsampling and aggregation empower your organization to handle increasing volumes of data without sacrificing on outcomes. Downsampling allows historical metric data to be read with **better performance while reducing storage costs over time**. The basic approach typically involves retaining data at coarser intervals over time – for example data that is initially collected at a 15 second interval may get rolled up to 1 minute after a month, 15 minutes after 3 months, and 1 hour after a year. Downsampling strategy should be closely coordinated with retention strategy, which is discussed in the next section.

Aggregation of metric data can be used to reduce the cardinality you don't need and only store the data that is critical to you. For example, the majority of metrics data is generated at the resolution of a virtual machine. But ask any engineer and they will want to reason about this data at the resolution of a pod. What aggregation can do is pre-compute the data at the resolution of a pod and get it to the glass, instead of querying all of the VM data every time. It delivers the same results but with **better performance at a lower cost**: Everyone wins.

Retention and resolution

To **combat data growth while containing costs**, your organization must make sound decisions based on business value about what data you keep, for how long, and at what resolution. Yet there are so many different use cases for monitoring data in modern cloud native environments, and each has its own set of requirements. For example, a user may want to view the per container metrics and break it down by the pod UUID as it's critical for deployments. This data will be extremely high cardinality and useful in real-time, but becomes less useful over longer periods of time — after a year, no one will care about the specifics of a particular pod since it most likely will not have existed for very long. Alternatively, specific aggregated metrics are useful for trend analysis over time.

The wide variety of monitoring data use cases makes setting a single data retention and resolution policy across all of them unwise. It's prudent to select an observability solution that gives you the flexibility to tailor and assign different retention times — such as both the time interval (15 second scrape interval) as well as the retention time (six months) — and resolution policies for different subsets of your monitoring data, depending on the use case.

Open source compatibility

Proprietary formats not only make it difficult for engineers to learn how to use systems, but they add customization complexity. Modern cloud native observability solutions natively integrate with open source standards such as Prometheus and OpenTelemetry, which **eliminates switching costs**. In times of economic uncertainty, and when tech talent is scarce, you'll want to invest in a solution that is open to possibilities.

WHAT TO ASK VENDORS ABOUT COST OPTIMIZATION AND RESOURCE EFFICIENCY:

- What capabilities do you provide to help tame data growth and contain costs, particularly as we expand our cloud native adoption? (e.g., rate limiting by team or label?)
- How does your solution dynamically adjust the resolution of any subset of metric data?
- How will I know if my teams are generating the greatest volume of metrics and cardinality?
- What kind of cost-attribution metrics are available?
- How do we create granular alerts and custom dashboards?

Gain consistent availability and reliability

Traditional APM and infrastructure monitoring systems can become unreliable — losing data or causing downtime — under the strain of increased data volumes and cardinality. No matter where you are on your cloud native journey, your business can't afford to be flying blind without observability to inform real-time issue response. Be sure to investigate how consistent availability and reliability are when assessing cloud native observability solutions.

Service-level indicators, objectives, and agreements

You should not be afraid to ask about modern cloud native observability solution vendors' commitments to system uptime and how successful their site reliability engineering (SRE) teams are at meeting them. This includes:

- **SLI (service-level indicators)** – The actual numbers measuring the health of a system.
- **SLO (service-level objective)** – The vendor's internal goals for keeping systems available and performing up to standard.
- **SLA (service-level agreement)** – The vendor's commitments (often legal) to its customers about system availability, response time in case of issues and the consequences if those commitments aren't met.

Ideally you'll want to look for at least 99.9% uptime SLA. In addition, find out what the actual delivered uptime has been for the past 12 months. Beyond the numbers, take time to understand how each vendor defines and monitors its SLAs, and at what point it notifies customers of problems. A best-in-class solution will proactively monitor its own systems for downtime and count any period greater than a few minutes of system inaccessibility as downtime, prompting immediate customer notification.

The vendor of a best-in-class solution will also have a cultural focus on SLIs, SLOs, and SLAs. That includes making internal SLOs for availability, performance, and correctness across core product functionality for all data types (metrics and traces) public. This ensures it is answering important customer questions for users including, "Can the service be used and trusted?", "Is the service fast enough?", and "Does the service return accurate results?"

SLA checks

Proactive SLA checking can tell you a system is working — returning 200s successfully — but a simple ping check against an endpoint doesn't reveal much about whether the system is performing as expected. A proper SLA check for a hosted cloud native observability solution should assess basic read and write paths to give you confidence that your data is persisting as expected and that none of your data is lost.

Dedicated endpoint with no noisy neighbors

Making sure metrics can be effectively queried without negatively affecting individual user experience is as important as guaranteeing writes. So look for a solution that guarantees your queries won't be interrupted by sudden changes in the workload or by "noisy neighbors." There are other customers in a multi-tenant SaaS environment whose demand surges reduce performance for other tenants. You can avoid noisy neighbors by choosing a solution that provides only a dedicated tenant, or endpoint, for each customer.

Cloud provider choice/circular dependency protection

Engineers understand the circular dependency concept — when A depends on B, but B also depends on A — but few consider this in the context of their monitoring service. If your production applications are hosted in the same cloud provider and region as your SaaS monitoring solution, you've created a circular dependency. The primary risk is a service disruption in that region: You could have an outage of both your production environment and your monitoring solution simultaneously. This would be a worst-possible time scenario to incur a monitoring solution outage — this is your system informing you of your production outage.

As a best practice, choose a modern observability solution that enables you to host your SaaS solution in a different region than your production environment, AND with a different IaaS cloud provider if possible, to avoid this circular dependency.

WHAT TO ASK VENDORS ABOUT AVAILABILITY AND RELIABILITY:

- What is your uptime guarantee?
- How do you define downtime and what solutions do you use to monitor for downtime?
- In the event of downtime, how quickly do you notify customers?
- What is your actual delivered SLA over the past 12 months for all customers?
- What is your actual delivered SLA for the top 10% of customers (by size)?

Increase engineer productivity

Most APM tools were introduced more than a decade ago when most engineering teams were organized in a top-down hierarchical fashion. As more engineering organizations adopt DevOps and platform engineering as disciplines, traditional APM tools struggle to support how modern engineering teams develop, deploy, and support their applications and infrastructure. As your business builds software by organizing people into small, interdependent engineering teams with focused responsibilities on certain services or features, you need an observability solution that supports that model. Yet existing APM tools don't reflect the reality of how your teams are now organizing, or the relationships between them.

Today's engineering on-call shifts are stressful because people can't find the right data, run queries quickly, or remediate issues fast. In a DevOps world, developers own responsibility for the operations of their applications. That's why any observability solution you choose needs to support workflows aligned with how your distributed, interdependent engineering teams are now operating.

Contextual views

Your observability solution should reflect how modern software companies organize their services and people. Unfortunately, most APM and infrastructure monitoring tools were built with the assumptions of monolithic applications being run by dedicated operations teams, versus today's reality of DevOps teams operating distributed systems.

Modern cloud native engineers need a way to easily navigate through observability data, gaining context and the power to easily zoom in on the data most relevant to the services for which individual engineering teams are responsible. They need to be able to quickly jump between data types and understand service dependencies. Any solution built before the cloud native generation won't be able to achieve this.

Query acceleration

Engineering time is valuable. To reduce idle time and improve your engineers' productivity, a best-fit observability solution will deliver an optimal dashboard experience and faster query response. Engineers should not have to waste cycles trying to improve slow queries or even thinking about how to write an optimal query. A best-in-class observability solution will detect slow queries and speed them up by creating a pre-aggregated time series of the data point to ensure faster performance. It will look for similar queries across all dashboards and use the faster, pre-aggregated version instead. Taking it one step further, these solutions can show executives the number of optimized queries, engineering hours saved, and other pertinent metrics.

Deep integration of telemetry types to avoid context switching

Metrics inform an engineer of a problem; tracing tells the engineer where that problem is, and by digging into the logs the engineer can find out exactly what caused the problem. Yet today's APM and infrastructure monitoring tools are too siloed and too noisy. When an alert fires, the engineer must investigate it, then switch over to distributed traces to triage the issue. Unfortunately, even when these tools are all from the same vendor, they are not deeply integrated and slow down the engineers with context switching. Look for observability tooling that offers features like deep linking and trace metrics to speed up MTTR (mean time to remediation).

Trace metrics, for example, accelerate the discovery of where an issue is because the top span presents the highest error rate — no more searching through every span. With a modern cloud native observability solution, it's now possible to define metrics based on traces returned from a query. A trace metric enables your engineers to generate a new metric data point based on an entire trace or part of a trace. The metric can then be used to create an alert and/or a dashboard.

Query scheduling

It should be impossible for one person to generate a large amount of query work that monopolizes system resources and prevents others from querying. In the same vein, a system should be prevented from being overloaded by the query work of a single user or many concurrent users.

A modern cloud native observability solution rate-limits reads in a way that does not unfairly impact any of your users, and protects system queries like alerts from being impacted by user load. It ensures that query resources are fairly shared by your users so if one issues an expensive query or a large volume of smaller queries, the database will break the work into chunks that are scheduled alongside work for other user/system queries. While many queries from one user may take longer, other users are not unfairly penalized, and the system's ability to evaluate alerts is undisrupted. That's something cloud native observability can do that existing APMs simply don't.

WHAT TO ASK VENDORS ABOUT INCREASING ENGINEER PRODUCTIVITY:

- How does your solution support the ways modern engineering teams need and want to work?
- How do you prevent disruption to engineering workflows and thought patterns?
- How are you helping engineers spend less time searching for the data they need — during triage or while monitoring services they regularly oversee?

Deliver world-class customer success services

Enterprise-grade service and support

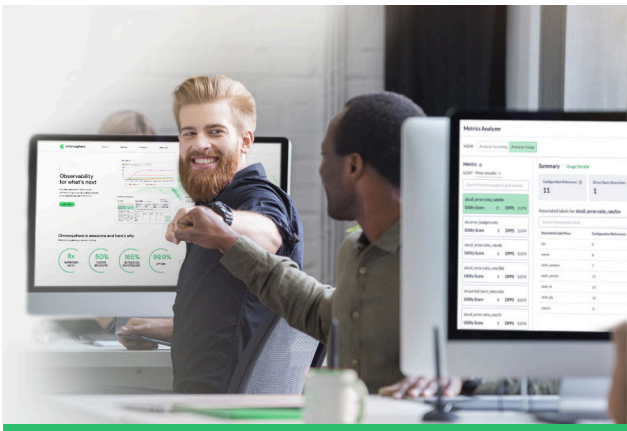
Although core technical capabilities are the primary consideration for choosing a new solution, you shouldn't overlook the services, support, and pricing offered by cloud native observability vendors. They can be game changers, particularly if you have strict SLAs or want a dedicated success team. Look for a company that's interested in being more than a vendor; that's a trusted, strategic partner, bringing expertise to your environment; and offers the agility and skills needed to help you navigate the unexpected on your cloud native journey.

Take note that many vendors will charge a significant fee for these services, so ask upfront **what it will cost you to gain the success-services that you need**. Consider the services you might need — migration from legacy tooling, on-boarding of historical data, assistance setting up alerts and dashboards, enablement sessions with users and administrators — and team with a vendor that can provide them and more.

Training and enablement

Training and enablement goes hand-in-hand with success services. You need to ensure that all of your engineering users are, at a minimum, proficient in the tool. Look for a vendor that not only gets your staff onboarded quickly, but offers continuous learning.

For many organizations, open source software such as Prometheus, PromQL, OpenTelemetry, etc, will be new to them. Look for training and enablement that not only teaches the core features of the product but also includes a curriculum that helps engineers build their OSS knowledge and skills. Example courses could include instrumentation with Prometheus and OpenTelemetry, building queries with PromQL, and migrating dashboards and alerts.



WHAT TO ASK VENDORS ABOUT CUSTOMER SUCCESS SERVICES:

- How many customers does each customer success representative manage?
- What is your approach to on-boarding historical metric data?
- What on-boarding services are available?
- What type of assistance do you offer for setting up alerts and dashboards?
- What enablement models and sessions do you offer for users and administrators?

Checklist: Cloud native observability

Capability	Key Features	Vendor
Tame data growth and control costs	Control plane	Yes | No
	Visibility and quotas	Yes | No
	Downsampling and aggregation	Yes | No
	Retention and resolution	Yes | No
	Open source compatibility	Yes | No
Gain consistent availability and reliability	Service-level indicators, objectives, and agreements	Yes | No
	SLA checks	Yes | No
	Dedicated endpoint with no noisy neighbors	Yes | No
	Cloud provider choice/circular dependency protection	Yes | No
Increase engineer productivity	Contextual views	Yes | No
	Query acceleration	Yes | No
	Deep integration of telemetry types to avoid context switching	Yes | No
	Query scheduling	Yes | No
Deliver world-class customer success services	Enterprise-grade service & support	Yes | No
	Training and enablement	Yes | No

Take a test drive

Now that you've completed an evaluation on paper, the next step is to take a test drive of the best-fit-for-your-organization cloud native observability solution. Typically, vendors offer a free pilot so you can test a solution's capabilities with your actual data set and ensure it meets your needs before purchasing. Pilot success depends on you having a clear set of objectives and results in mind. This should include how and when you are hoping to scale your cloud native model. Also be sure to ask about the solution's:

- **Pricing model** – When it's difficult to compare vendor pricing, focus on determining if the cost is fair and easy to understand. It should be predictable and only grow as your value from the solution grows.
- **Security** – To ensure your data will be well protected, be sure to discuss access permissions and whether administrators can prevent unauthorized persons from making system changes. Also remember the security advantages of multi-tenancy with a SaaS service is such that all of your data is completely isolated from other customers'. This makes the possibility of cross-account contamination, or other types of breaches, extremely low.
- **Management overhead** – Not all solutions are complete end-to-end offerings. Additional management overhead can prove to be time consuming and expensive as well as create single points of failure and lower availability/reliability.
- **SLA track record** – Know what a vendor's historical performance is on SLAs as well as what SLA it's promising in the future. Remember, too, that SLA definitions vary by company, so ensure you understand how vendors define an outage when you make side-by-side comparisons. You'll want to ask not only for overall customer SLA performance, but also for the SLA performance of the vendor's biggest, and likely most demanding, customers.
- **Reference customers** – As with any vendor selection process, speaking in person to existing customers is a key part of the decision-making process. If you have specific concerns, it's an especially good way to hear how a particular vendor has addressed them with another business.

These additional criteria can help you make an even more informed decision, particularly if you are piloting more than one option.

As you consider solutions for the scaling of your cloud native approach, you may come across Chronosphere. That's because it's the only purpose-built, SaaS monitoring solution for cloud native environments, providing deep insights into every layer of your stack — from infrastructure to applications to the business. The Chronosphere platform reduces customer observability data volumes by an average of 48% while improving key metrics such as time to detection and time to remediation. It delivers capabilities that benefit central observability teams and makes the lives of engineering teams easier by streamlining workflows, accelerating remediation, and improving both engineer efficiency and quality of life. In real-world customer environments, Chronosphere is reducing data volumes by 89% and giving organizations 99.99% availability. Request a demo at **chronosphere.io.**

The right cloud native observability solution can deliver tremendous competitive benefits to organizations like yours, whether you're just starting out or have multiple applications in production. Reimagine your solution and processes to become an elite performer.