Kong

EBOOK

Unlock App and Service Performance With a Decentralized Architecture

Content

Is your organization ready to decentralize? 4
Challenges of centralized architectures 6
Slower time to market 6
Keeping pace gets harder 7
Pileups: Kubernetes on a legacy system 7
Monolithic bustle delays decentralization 8
Poor end-user experience 8
Compliance turnaround 9
The coming “data tsunami” 10
The superhero problem: Securing centralized DevOps grows Herculean 11
Security complexity at an exponential scale 12
Progressing threat sources for legacy systems 13
Key challenges to decentralizing 14
New costs 14
Manpower availability 15
DevOps downtime 15
Technical fears 16
How to decentralize applications and services at scale 17
Tech-agnostic decentralizing strategies 18
Choose the right infrastructure 18
Carefully define the API 18
Manage the platform 19
Automate what you can to accelerate services 19
Exploit the interruption 20
Conclusion 21

Introduction

Ask development teams about their current worries today, and you may get some pretty unique answers. There are so many complexities in a given DevOps environment that any one answer is bound to be incomplete. There are some questions, however, which are at the forefront of every development team's concerns at present.

Whether or not to decentralize applications and services is such a key concern for many enterprises. Apart from various impinging outside issues like industry economics and network security concerns, there is the immediate bottom line need for greater productivity on current and pressing projects. Decentralizing promises faster speeds, greater options and the ability to accelerate your sheer development capacity overall by exploring microservices and related capabilities in today's DevOps world.

In this eBook, we will attempt to address the key problems and solutions surrounding the choice to decentralize applications and services, beginning with readiness.

Is your organization ready to decentralize?

As our Digital Innovation Benchmark 2020 reveals, 84 percent of enterprise organizations have already embraced microservice-based applications. Another 40 percent of those surveyed stated they believe that staying ahead of the emerging technology curve is central to growth within their industry by 2023. According to those surveyed, distributed architectures are an established part of the curve businesses must adapt to. That said, development teams can't reap the benefits of a decentralized DevOps strategy without first having adopted a more distributed architecture.

Traditional API management can't keep pace with the mushrooming API universe.

Within the ever-morphing requirements of an increasingly connected, complex world of computing, acceleration of software delivery has become a top business priority. However, heightened legacy API management clutter creates a resource drain that stalls or even halts the team's ability to meet that acceleration. This results in numerous and often challenging problems for

development teams to overcome. With all of the bustle, gaps in application and environment security become overlooked, and these issues are multiplied on legacy runtime environments.

In this vein, Gartner has noted that there is a renewed interest among enterprises to establish fully functional and expanded in-house DevSecOps centers. Moreover, a recent survey conducted by Ponemon Institute reveals that over half of enterprises polled reported that their own SOC (security operations center) did not adequately align with their own needs. Quite simply, the challenges are too difficult to figure out in short order, while trying to keep an eye on the bottom line.

Challenges of centralized architectures

Centralized legacy architectures may be easier to secure stored data due to being localized, but the value there tends to end rather sharply and abruptly for developers. Without upward productivity potential, software development environments can't meet the needs of the market to push the brand and its products forward. By compounding such cumulative development difficulties, the problem morphs from "common development snags" into wide-reaching business problems that ultimately affect the bottom line.

Slower time to market

One of these challenges is a slower time to market. Centralized resources depend upon the limited speed and variability of a single monolithic architecture. Complications of this methodology arise from lag and latency, unpatched aging software, and security gaps. Experience shows that this results in more time spent repeating the same things and often doing them worse.

Distributed architectures allow faster delivery of applications and services, accelerating product launch. The sooner the product is launched and stays functional, the easier it is to provide customer service. Additionally, this creates more space for developers to focus on the next task, rather than languishing too long in maintenance mode.

Keeping pace gets harder

Keeping pace becomes more and more difficult under legacy restraints – and not just from being centralized in a decentralized universe. Configuration inconsistencies naturally multiply under the conditions DevOps teams find themselves immersed in today. Leveraging a monolith architecture without real time authentication checks and balances gives way to frequent application failures, which further slows the delivery of business value.

There quickly arises a need to reroute traffic to get around overtaxed resources. In order to get microservices from point A to point B on time within a given service, you'll need them to travel unimpeded. Workarounds begin with custom coding, and the ever-slippery slope of lost production time pushes development teams even further behind.

Pileups: Kubernetes on a legacy system

Kubernetes can be adapted to support a monolithic architecture, but it's unable to effectively manage legacy API gateways. Among these management issues is the need for a single truth-source for Kubernetes, which is difficult to configure on a monolithic architecture. Moreover, difficulties arise from inconsistencies among application components struggling to keep up in real time.

Pile-ups are inevitable here, but what they breed is exhaustion. This creates a situation where one less efficient method requires yet another just to keep up with work that newer methods can handle out of the box. All this rigging work is done only to accomplish similar tasks ever more slowly.

Security loopholes grow from legacy ad-hoc approaches to Kubernetes as nothing fits seamlessly. Furthermore, each constituent component on a legacy system may be custom-made by entirely different coders with rushed or scant existing documentation. Keeping clarity on the runtime environment, let alone the application, will be daunting.

Monolithic bustle delays decentralization

As roles and teams expand in a development environment, fresh problems arise: audits become harder to execute, and time crunches become more frequent for production teams with more in the backlog.

Time lost doing things in arcane ways inevitably means lost opportunities. Issues for legacy systems include app unreliability deriving from changing hands many times over and using older software that's no longer supported. Shrinking expertise over languages that are no longer common creates shortages in badly needed team resources. Outdated or incomplete documentation equates to outdated best practices for the technologies at hand, as well as increasing the difficulty of onboarding new resources.

Building applications as constellations of microservices is par for the course in today's computing cosmos. Legacy API gateways are simply not designed to connect microservices within and across clouds and protocols. This realization causes teams to push microservices further off into the future. Even with the most robust and consistent homegrown coding approach, the native difficulty of connecting to required outside nodes merely exacerbates the problem of pace.

Poor end-user experience

Lackluster performance is common in legacy monolithic environments lacking comprehensive and automated service controls. The client and user suffer from being on the downside of the upward technology curve. Relationships are impacted alongside performance as both company and customers struggle to cope with disorder.

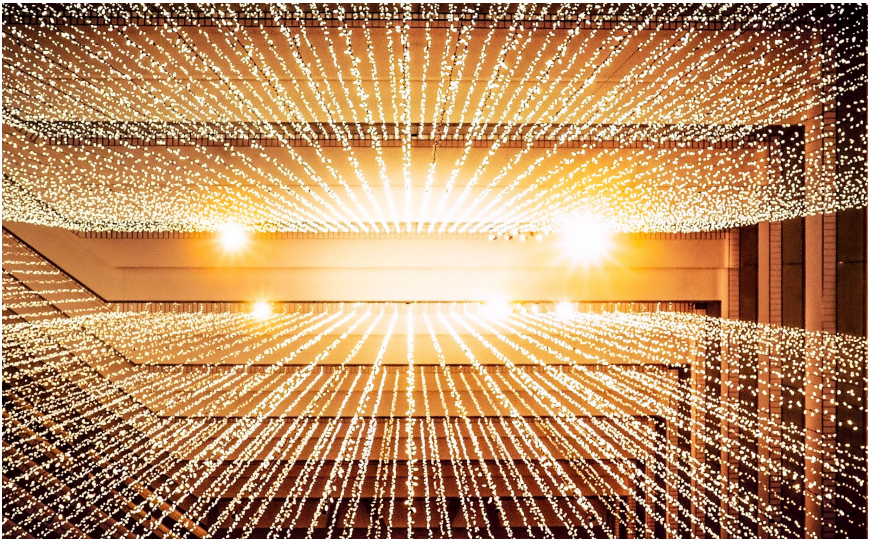
Upward scalability is close to the heart of every business, but scaling easily becomes hampered due to increasing complexity of existing services. This may seem counterintuitive, since more services should mean more business, productivity, and ultimately, growth. Latency and security vulnerabilities can slow or even pause service delivery, even with a partially distributed system in play. The app stops working. Transactions are interrupted. Business grinds to a halt.

Compliance turnaround

Risk management and due diligence inevitably get compromised from over-complex solutions to simple problems. Compliance audits become harder to execute and thus compliance standards become dangerously difficult to maintain, while speed and agility suffer under normal work loads.

More people won't solve monolithic compliance woes. At a certain threshold, more people isn't a viable solution. The more intuitive answer is a less troublesome constellation of technologies. Data-processing requirements don't shrink, but time and human resources inevitably do. Under a monolithic legacy configuration, services are increasingly called to use siloed data that benefits from more isolation, which the monolith simply cannot support.

The coming “data tsunami”



Within our current technological model, we can safely anticipate an unpredictable onslaught of new varieties of data sources entering the digitized flow every quarter, every week and even every hour. These emerging sources are typically automated rather than human-made. At some point, the data is simply too overwhelming for any one monolithic architecture to keep up with.

Noteworthy examples of exploding data sources include new technologies and expanding uses of current ones, such as IoT and smart devices that will multiply the data ecosystem exponentially each quarter, as well as ubiquitous upgrades to high-speed internet and true 5G networks which invite more media consumption. Included are digitized TV advertising and open source tools that make AI automation of everyday tasks accessible from the web.

Additionally, arising datasets for predictive analytics and pattern recognition play a role in complicating a centralized

approach to data as do emerging innovations such as car-as-a-platform (CaaP) and an ominously staggering rush of quantum computing data inputs which must be decentralized. Here, we enter the need for a decentralized app (dApp) that can pull from other distributed resources quickly and efficiently, taking the burden off monolithic systems and resources by boosting microservices, while being able to compliment local computing resources that handle sensitive company data still held closely.

Since dApps rely on distributed architectures to handle a massive amount of processes, there will be less and less room for centralized architectures as dApps become the standard. Blockchain will be used increasingly to secure the transported data, further excluding or slowing the pace of a centralized environment.

All this adds up to a much higher demand on resources to keep pace with emerging changes in application development, data storage and platform-as-a-service (PaaS). With this new model of emerging data sets, it becomes clear that there is a steep cliff somewhere ahead in the monolithic mists – a point at which continued progress is interrupted. At this point, there are very few options, and the path could be a blinded one. It doesn't have to be.

The superhero problem: Securing centralized DevOps grows Herculean

The superhero problem is just a way of saying that counting on super-capable heroes isn't very strategic for a development team or for monolithic-heavy DevOps. When taking all of the preceding monolithic challenges into account, SecOps complexity can be seen rising at a frightening scale without any realistic expectations on risk containment. Superhuman prowess becomes mandatory, and of course, is increasingly impossible to manifest.

Security complexity at an accelerating pace

Time and coordination become more sensitive while resources become scarce in this so-called "superhero" scenario. As the re-configuration of systems grows more unwieldy and ad-hoc, compliance issues pile up beyond the merely human on a monthly or even weekly basis. The welcome prospect of automation appears as an answer, but at what costs to production? The temptation can be to stay in survival mode, just treading water.

New east-west data traffic raises the bar for SecOps. Virtualized servers have their own vulnerabilities. Siloed data that interacts with decentralized architectures is already exposed to the outside. More services are now needed for each new type of data. All of that emerging internal bustle requires an automated, real-time threat baseline, as well as automated monitoring and detection capabilities that are increasingly pulled from offsite.

Unavoidable legacy interactions with the outside become whack-a-mole on steroids. Security policies for an expanding universe of microservices are no longer viable for monolithic systems. If anything, it's now whack-a-mole at the microscopic level, multiplied over each new service and microservice for each new and existing application, as threats and virus updates become more complex and more cunning. New exploits simply multiply the total threat surface.

Automated hacking has changed the security game. Combine all the above scenarios with the automation of viruses that machine-learn within an AI-driven hacking culture, and you now have a situation where your own legacy host resources can become zombie servers, draining computing power and endangering precious virtual cargo.

Needless to say, while the adaptation to the tech curve can be maximized with fresh, sudden capital inputs and emergency stop-gap measures, the ride will be arguably bumpy and not infrequently climactic. Automated monitoring is the only way to fight back.

Progressing threat sources for legacy systems

- **Legacy systems grow more likely to introduce new software**. The alternative is to write it all. New pre-packaged code will continue to ship with already existing loopholes and required patches that legacy systems can't truly cope with out of the box. This automatically hikes costs, security and compliance risks far beyond those for a decentralized system in the same scenario.
- **Unavoidable, disadvantaged involvement with evolving cloud-centric exploits** further puts legacy systems at risk. These increased rates of threat and attack incidents will demand more security vigilance than is possible without cloud security in place. Response delays will put home resources at a distinct disadvantage.
- **The future portends blockchain data security protocols within transactions.** That effort depends upon an advanced, nimble, cloud-centric model of computing. As the SecOps community continues to examine the ways that blockchain's introduction will upgrade security, one step beyond that sees quantum blockchain as a natural progression. Neither blockchain nor quantum are manageable technologies for a centralized architecture.
- **Correcting AI bias in the cloud is fast becoming a normalized concern.** This will demand an arrangement of resources that can deal with high-speed interactions, as well as requiring decentralized resources to run those interactions effectively and safely during security crises in progress. Not being able to take advantage of AI bias correction will become a liability for development teams and end-products over time.

In all of the above scenarios, a cloud security solution working across decentralized applications, services and their constituent microservices will be faster and more effective for the organization. For legacy centralized architectures, however, the response will tend to be so slow and inefficient that sometimes simply abstaining from a security practice will be the inevitable default. In some cases, a centralized system will simply be unable to cope with securing services or microservices in real time. Security from partners will mean little if on your own end the security is breached or not viable.

Key challenges to decentralizing

Obviously, there is a trade-off for upgrading application environments toward a decentralized model that can easily fend off today's revolving threats and preserve service and session integrity. Weighing and balancing these trade-offs is time-consuming and therefore risky in itself. A clear outlook on the technology curve is essential.

1. New costs

New cloud resources carry unknown security risks. This is a compliance and risk management issue, which costs time, money and other resources to oversee. New expertise and outside personnel could be required. Capital allocation may require more assurances to stakeholders as well.

Compared with the many costs of not decentralizing, treading water doesn't add up. These include security, risk management, compliance issues and lost productivity – hidden costs of a legacy system budget, not just for now but also for the future.

2. Manpower availability

SecOps for a decentralized architecture may inevitably require more specialized teams on a regular basis. There is also the issue of lost coding time while conducting in-house training. These will result in some lost productivity if outside help can't handle low-level tasks in the interim.

Ramp-up ultimately requires only short-term personnel substitution. The alternative is an ever-increasing shortage of hands to swat ever-increasing security threats from all corners of a centralized island, where even stored, non-moving data isn't really truly safeguarded.

3. DevOps downtime

DevOps downtime in the short run is always inconvenient and puts off launch. As we've already seen, the centralized architecture often seems to have ample reasons to avoid development delays of any kind. They already operate at a marked class below what is becoming the norm.

The short-term hiccups in current projects will be forgotten quickly. It's worth short-term expense and inconvenience for the sake of a much more robust development capacity in the long term. SecOps will not really become a downtime factor for DevOps but rather promises greater predictability and a new level of reliability, ensuring projects are protected and can launch with a specified due date. DevSecOps as a whole becomes a consolidated bicameral system over time as the two halves begin to dovetail with each new micro-upgrade.

IT overall benefits from the beefed up security protocols as well. Since these get instituted for the organization as a whole, there is more than just development risk at stake in staying put. The new norm for legal and compliance allows greater confidence of all stakeholders to move in new directions.

4. Technical fears

There is one more cloud of fear distorting this topic. According to Quali, key barriers that inhibit true DevOps from arising consist of: company culture (14%), testing automation challenges (13%), legacy infrastructure (12%), application complexity (11%) and budget limits (11%).

One of the many issues centralized servers now face is being tasked with running services or microservices from dApps for things like P2P networks, such as games, online storefront transactions or even interacting with smart contracts for blockchain services like virtual currencies. Increasingly, the centralized server is expected to compete against decentralized servers offering the same microservices, and in this aspect, they are at a distinct disadvantage. In many cases, the teams involved may have a problem imagining their own processes dealing with such issues under any conditions.

New and existing strong talent can dispel the technical storm clouds. An understandable lack of in-house expertise is a crucial factor stopping many organizations from decentralizing into a more distributed architecture. The idea of cost suddenly reappears when centralized cultures dream of going decentralized. There is a watershed moment when the cost problems of one paradigm dissolve into the new paradigm's possibilities for talented individuals.

Hiring one or two leaders in charge to decentralize changes the vibe. Remarkably, existing talent is also able to shine all the brighter once they're no longer expected to run the crumbling monolith game. Cost concerns and talent shortages both dissolve into that new level of capability and the new opportunities it brings. The storm clouds soon turn into white fluffy clouds of sunny optimism, or at the very least, time is no longer lost in fearing the unknown.

How to decentralize applications and services at scale

Decentralizing at scale will be essential to the success of many organizations looking for new, decentralized freedom. The following strategies can guide teams through the goal posts to a clear win for all. But first, a word about mindset.

Tech-agnostic is key. Avoid the persisting inertia of former lopsided monolithic architecture reliance to help address all of the issues and threats we've covered thus far. Learn to see a new norm forming – one of new possibilities.

Constantly evolving technologies like serverless and Backend-as-a-Service (BaaS) are breaking up monolithic approaches into a free-flowing, multi-technology universe where everything revolves around a technology-agnostic sun. Each new project or problem will require new approaches, new kinds of solutions and new ways to secure it all against a fully up-to-date threat baseline that is not just frequently updated but *live*.

Tech-agnostic decentralizing strategies:

1. Choose the right infrastructure

A general best practice for decentralizing is to take additional steps to also move toward a distributed, cloud-first but also cloud-agnostic approach. Decoupling the monolithic application facilitates breaking up services into microservices that can shift the focus from managing services toward managing a self-monitoring DevOps culture. It allows the organization to adapt to changing needs and requirements in the marketplace and to adopt suitable emerging, sometimes game-changing technologies like blockchain. Additionally, companies are then allowed to deal more effectively with vendor problems or market disruptions with more flexibility ahead of the curve.

If deciding to opt for a branded cloud solution chain, understand that the future will feature this recurring crossroads and continually present few reasons to employ a varied set of solutions. Sometimes it can be delayed when business goals don't align yet, but staying open to the future is most helpful to a strong decentralized approach going forward. Being cloud-first is still a step closer toward hybrid-cloud.

2. Carefully define the API

Thoughtfully defining the API is a first step for making applications and services work better in a decentralized world. Talk to the client of the API to understand its use case and ensure the API will address its needs. Is the consumer outside your organization? How sophisticated is the consumer at API integrations? Avoid bloat and too wide an array of extraneous purposes for the API, as any bloat will ultimately limit productivity for key processes and their clients. Consistency in binary or similarly logical naming protocols can also dramatically reduce confusion and pointless referrals to API documentation later in the game.

Whether or not you're starting from ground zero, consider that app components are best architected, developed, stored and moved together whenever possible. A change to one demands a review of how the others will work with those changes. So above all, your application protocols should be 100 percent coherent.

3. Manage the platform

Reviewing end-user personas will yield helpful insights to ensure that APIs stay focused. DevOps, SecOps, developers, testers and stakeholders should have the same top-level grasp of the underlying technologies so they can make decisions together with greater ease. Periodically review the platform, question its integrity, optimize and harden any suspect areas accordingly.

Create a single source of truth in Kubernetes to better manage containers and the full lifecycle of your APIs through a Kubernetes-native interface that enables a highly consistent and dependable level of operations across clouds and infrastructures. Moreover, you'll want to configure the gateway and Kubernetes from the same place, preferably using custom resource definitions (CRDs) to handle both with greater ease.

Employing enterprise-class life cycle management tools is central to most of today's more advanced decentralized DevOps cultures. Ensure that the core is exposed only in functional ways that are decodable only to the clients themselves. Avoid indicating resources within protocols that don't help your immediate cause or that could help code intruders. Beef up governance policies to lock in consistent security practices to bolster confidence in running a much faster platform than before. Look for ways to avoid being pigeon-holed into any one technology. And above all, only choose a platform that allows all of this, if and whenever possible.

4. Automate what you can to accelerate services

Contemplate the entire software development life cycle as a whole. A lighter-weight open source proxy can help dramatically reduce latency, speeding services and microservices, but only an enterprise-class service connectivity platform can facilitate higher, competitive levels of automation. With that, you can abstract functions into the gateway or services control layer and take advantage of platform plugins to make short work of authentication, traffic control, transformations, and tasks like logging and metrics. Make sure your enterprise connectivity platform provider provides a plugin development kit (PDK) to help simplify any custom coding.

Make DevOps automation-building your new focus. On the DevOps side, instituting a more seamless CI/CD toolchain will reduce the downtime between project phases by helping everyone stay on the same page during development. It also helps when streamlining delivery processes.

Intensified and streamlined cross-training for DevOps and DevSecOps as a whole will benefit production and security teams, project lifecycle and stakeholder concerns, while supporting the automation habit. A robust automation strategy should ultimately be aimed at not just faster solutions but also vastly improved performance ratios and efficiency, continuous improvement that is coupled with improved customer experience, fewer rollbacks or system failures, and help to raise ROI via key performance indicators that matter to all stakeholders.

5. Exploit the interruption

There is no better time for upgrading your IT culture than when upgrading to a decentralized architecture overall. Let's face it, it's not easy reserving time for a complete systems overhaul. When you do, wisely use the excuse to make room for other worthwhile and aligned upgrades. This makes it far easier to implement all of your system changes within the same timeframe and with the help of the same teams, with team members still fresh on all of the topics involved.

It may help to use the tail-end of your upgrade project to review this process to help deliberate on and introduce new automation goals in the future. Do the same for the required DevSecOps regimen to sustain the new productivity model and standardize process targets. Everything can be a learning experience.

Conclusion

The application and services ecosystem is evolving under our feet. Playing catch-up is not a viable plan after 2020. Legacy monolithic systems that can adapt, upgrade and do plug-and-play with the cloud will fare better than those who can't.

Decentralized applications have fast become the single-most dynamic factor in the new normal. Industries are being disrupted and rocked. Blockchain and BaaS are becoming more frequently leveraged, and this innovation is all currently happening within and surrounding decentralized applications. Cloud-agnostic strategy is becoming a definite bonus to stay competitive. All of these recent trends depend upon utilization of a decentralized architecture that is easier to deploy and on which services and applications are easier to defend.

A great many currently rising IT industries make use of the decentralized, tech-agnostic model to achieve results and dominance in their own respective fields. These ultimately indicate the coming need for appropriate decentralized systems to plug into quantum computing to solve monumental, ever-new kinds of problems. Existing services must interact easily with these far-flung architectures in order to best serve their API clients and thus their end-customers.

Investing in the proper security and SecOps for these high-stakes industries will necessitate the use of decentralized approaches. This is the best way to serve the API client and the end user. The rewards include growing productivity, more fully exploiting market share, and reducing compliance issues and risk to a minimum.

In the final analysis, the goal whenever decentralizing is to find a way to meet contemporary requirements for scalability with the most agile and robust service connectivity platform available. It also requires an organization to answer the call for high-performance APIs with a correspondingly consistent low latency. Lastly, it obliges businesses to pair the demand for ever-greater levels of flexibility with a fully extensible, deployment-agnostic approach for a more predictable business model.

Kong is currently the only platform that connects every type of deployment, without actually limiting the user to any one particular technology silo. From monolith to microservices, from Kubernetes and mesh to hybrid cloud and emerging patterns of deployment, using a solidified and coherent approach is absolutely essential to a successful transition toward leveraging fully decentralized applications and services.

Konghq.com

Kong Inc.
contact@konghq.com

150 Spear Street, Suite 1600
San Francisco, CA 94105
USA