Serverless platforms and cloud strategies

Serverless Architecture Whitepaper

Dive into the new role of serverless with 8 articles and interviews from our experts in the field of the cloud and serverless architectures.

@ServerlessCon  # SLA_con

serverless-architecture.io
# Contents

## Serverless Development

How to support and scale digital experiences – the role for serverless CMS
Considering the serverless approach
By Scott Massey

“Serverless future will depend on the serverless orchestrator”
New kid on the block
Interview with Veselin Pizurica

“We are in the early stages of a huge shift away from building SaaS product capabilities in-house”
Big changes for Software as a Service
Interview with Sagi Rodin

## Serverless Architecture & Design

Strategies for the Cloud Threat Hunter
Collect data, analyze, and respond
By Asaf Shahar

“Lumigo is purpose-built for cloud-native”
Making sense of cloud-native complexities
Interview with Erez Berkner

A Cloud Data and Analytics Platform Has Major Benefits, But That Doesn’t Mean it’s Easy to Manage
Find the right mix of value to the business and value to IT
By Dave Taddei

No-code is SaaS 2.0
Moving towards no-code/low-code
By Jag Lamba

## Serverless Operations & Security

“Jamstack allows developers to decouple the web presentation layer from the backend logic”
Current serverless model challenges
Interview with Matt Biilmann
Considering the serverless approach

How to support and scale digital experiences – the role for serverless CMS

Using a serverless content management system (CMS) should ensure that developers can concentrate on what they are there to do – write code. It can remove the burden of running back-end infrastructure and let other teams manage the content that is put in place.

By Scott Massey

Digital Marketing and Customer Experience have become very specialised and evolved as fields of study and career paths. Building and managing the digital experiences these teams design usually involves a team of developers creating and maintaining those assets. However, these developers often also have to create the infrastructure that power these experiences. Alongside the front-end digital experience, the back-end also has to be monitored and supported over time.

This all has an overhead for developers as it involves creating and building code and infrastructure, which will need support and updates to maintain relevance and impact. This is problematic when it requires double duty to understand infrastructure and the content that is displayed. Regardless of whether these assets might be simple text or complex digital experiences, they will still need support and upkeep. This can be a big source of overhead for developers, stopping them from focusing on coding and creating.

A content management system, or CMS, can vary from a basic website solution that will handle writing, formatting, and editing of pages through the administration panel of a website through to complex systems that have to serve different content and languages for tens or hundreds of sites. Alongside this, the CMS may also integrate with other functions, such as personalization or other MarTech tools.

One way to overcome this problem around developers supporting complex and disparate infrastructures, is by splitting the front-end systems away from a back-end CMS that provides the content for one or more websites. To implement this decoupled or headless CMS approach, developers will need to think about how it will operate over time, and how it will handle work for other teams. More importantly, they will want to hand over the management side for the infrastructure as well.

Looking at your CMS options from a long-term ROI perspective

Traditional CMS systems deliver the entire website, where the database that stores all the content is integrated with the front-end, where experiences are rendered. While this works, this one-to-one relationship between the data model and the display engine doesn’t scale to meet today’s demands, which may require multiple, sometimes hundreds of websites. Additionally, new devices and display options can more easily be launched outside of a monolithic CMS.
Moving to a headless CMS approach can help as developers can hand the responsibility for managing content over to other teams. Alongside the act of implementing a headless CMS in the first place, there is also the question of how that CMS should run over time. This is where a serverless deployment can be a good option to consider.

From a content perspective, all the data that the site may need over time is stored in the CMS. This can be accessed and consumed using APIs. For developers, using APIs is normally easier and more efficient for them compared to building site infrastructure manually. This approach also makes it easier to build new services or deliver more functionality into the site.

Implementing a headless CMS will involve designing a more thorough data model than a simple website might require. Also, the right deployment method and infrastructure components will need to be considered carefully. To start with, you will have to choose a mature CMS like WordPress or Drupal to integrate with, and where to implement the service. This will normally be in the cloud, as this can offer the scalability and management needed. Alongside this, your sites will have to integrate with a content delivery network (CDN) to speed up delivery of that material to users.

Adopting a serverless approach to this hands over the responsibility for managing that infrastructure over to the service provider. Their service should bring together all the components needed to operate and support the CMS, while the company gets to pay for what they use. For developers, this means they can concentrate on code, with the added benefit that the other teams involved also don’t have to take on the management tasks around the CMS.

Other considerations in migrations

There are other improvements that can be delivered using serverless. For example, any online service or site today has to have strong security. Moving to a headless CMS is no exception – this can increase your potential attack surface as you will have more moving parts to consider. Each component will have to be hardened before being moved online, and then kept up to date and secure over time. Platforms like WordPress are commonly targeted because they are so popular, so new issues are found over time.

For teams that have adopted DevSecOps principles, this should be an easier transition to make, as developers, IT operations teams and security are used to collaborating around common goals. However, for Website Operations, the responsibilities can be harder to lock down. When it comes to sites and software, developers can be held responsible for operating the whole site when this is not their usual remit. Similarly, non-technical teams like marketing can be held responsible by the business when they are either not well-versed in security, or don’t understand the necessary work involved.

Looking at Website Operations (or WebOps) can help in this process. Similar to DevOps, it is about making it easier to collaborate around projects that cross over between business and technology goals, in this case around site design, development and infrastructure. By getting teams to collaborate and understand the wider goals, each team can ensure they are fulfilling their part of the goal. At the same time, this can support more migration projects that simplify the overall process. As part of moving to a new CMS, developers can migrate things off their plates and then concentrate on where they can produce the best software and digital services.

Alongside this, using a WebOps approach can ensure that security teams are consulted and understand what is going on around any deployment. For example, if your organisation does not have expertise in hardening Drupal or WordPress instances, then handing this over to teams that do this all the time can help developers be more efficient and the security team to see that all the necessary steps are in place.

Using serverless CMS should ensure that developers can concentrate on what they are there to do – write code. It can remove the burden of running back-end infrastructure and let other teams manage the content that is put in place. It also provides a focal point for wider changes around the processes and requirements that take place around these projects, and can support wider WebOps adoption.

Rather than issues falling between different groups and becoming contentious, or being loaded on the developers because they are automatically assumed to be responsible, this approach can make these problems more understandable, and put the responsibility for changes where it should be. When you think about it this way, this infrastructure change can lead to more opportunities to improve everyone’s interaction, understanding and results.
We spoke with Veselin Pizurica, CTO and co-founder of Waylay about the serverless paradigm. What concerns do enterprises have regarding serverless adoption and security issues, how can they achieve monitoring and observability of serverless applications, and how will the world of serverless evolve?

Interview with Veselin Pizurica

Serverless Architecture Conference: Serverless is not the "new kid on the block" in the world of development anymore. Still, not that many enterprises have adopted the new paradigm already. Why is that?

Veselin Pizurica: It is correct that two different impediments are hampering serverless adoption. That is also confirmed and described in the O’Reilly survey on serverless: Concerns, what works, and what to expect [1].

Everything starts with a rather simple idea of developing microservices and connecting different cloud functions, but in most cases that leads to eventual architecture complexity. Therefore we often see this being manifested as a problem of tracing and observability. The architecture complexity is often accompanied by a complex deployment model.

The second hindrance is related to nontechnical concerns and fear: fear of losing control, fear of vendor lock-in, fear of weak security, unpredictable cost etc. Fear of (losing) ownership kicks in because you are delegating your application and business operation to someone else. Many big companies are offloading their workload into the cloud without knowing whether the total cost of ownership of the application will reflect their business goals. They will be charged per volume consumption, and hence their cost might skyrocket while their business is still in the early stages. As the cost is driven by the use of cloud microservices, cloud-native architects have also become responsible for the OPEX cloud cost.

Serverless Architecture Conference: You said that there are concerns regarding security?

Veselin Pizurica: Attack vectors by hackers can be coming from all directions, either by trying to get into the network via compromised devices, by hijacking the login credential of the person with access to the application or by exploiting a weak/obsolescent security stack in the application.

Therefore each of these threats needs to be addressed carefully. Security and identity are extremely important aspects of any software solution. I mentioned identity separately, even though it is often seen as part of the Identity and Access Management (IAM). The reason is that in the IoT world, proving the identity of a device is not trivial.

Having said that, LPWAN solutions such as LoRA and Sigfox are extremely robust in that sense. Another aspect of security is API integration with 3rd party systems. The trust and the access scope are often achieved by OAuth2 integration or via API keys, and in any case, very careful handling of these communication patterns and capabilities need to be provided by SAAS integration platforms. Finally, analyzing the usage patterns from outside and inside (by monitoring network traffic patterns, CPU load etc.) should be an integral part of any SAAS solution.

Serverless Architecture Conference: What about monitoring and observability of serverless applications?

Veselin Pizurica: One might ask himself how we got here in the first place? If the promise of serverless is to make things simple, by creating and sharing small snippets of code and mixing them with other services, how is that
everyone talks about traceability, debugging and observability as the major problem in serverless? The answer is deeply buried in the problem of stateless functions. For functions to be reusable, they must be stateless.

But overall logic, use of these functions is not, the context in which these functions are used needs to be captured. That is what we call a “stateful” part of the process. It might be about object relations, or a “pointer” in the overall process handling, whatever.

That is where observability comes into the picture, making sense of all these moving parts. In order to solve this problem, we don’t need yet another – better observation tool. Before we even talk about observation tools, we need a good orchestration paradigm, something that will “guide” these code snippets, something that can make sense of all these moving parts.

Serverless Architecture Conference: What are the benefits of such a high level of abstraction?
Veselin Pizurica: Let’s start with the downsides. Experienced developers and architects are always skeptical when they hear about yet another model-based automation platform. They feel that soon they will discover “gotchas”: instead of frameworks helping them with implementation, they will need to work around the limitations of the framework imposed on them.

There is a term used in computability theory called Turing completeness. If somebody says “My new thing is Turing complete” that means that in principle, it could be used to solve any computational problem. Software languages are Turing Complete. When serverless hit the mainstream, it was widely accepted that serverless was the best candidate for “the low code lego brick approach”. And that brings me to the story of Turing Complete automation. If we are to use code snippets to implement our logic, we need an extremely powerful rules engine that can orchestrate these code snippets – without needing to resolve back to coding all that in the software language, otherwise what’s the point.

At Waylay, we have created almost, if not Turing complete automation technology, which can orchestrate these snippets of code. That means you don’t have to code lower layers. We also want to liberate developers from stitching all these microservices together. That means that you don’t have to worry about infrastructure. Our ultimate goal is to liberate developers from getting bogged down in things that have nothing to do with the problems they ought to solve. The Waylay platform is a pre-made automation stack where API gateways, multi-tenancy, lambdas, databases, and all other services are embedded and pre-installed. Developers have nothing to set up, nothing to manage.

Serverless Architecture Conference: Serverless and Machine Learning – how are they connected?
Veselin Pizurica: There is already one very successful “marriage” in place, and it is called machine learning pipelines, for training and benchmark of machine learning models. In most cases, they are built using serverless triggers on new data arrival, like for instance setting S3 object store triggers, which would trigger post-processing serverless functions that might include executing model training or model validation.

In the wider context of automation, that is a less obvious path. When the model starts drifting do we retrain it or not? If that was used in the context of anomaly detection, what made that model not perform? Is this a data problem, is it a model issue, or real anomaly detected? When the same model is applied for prediction, let’s say energy consumption prediction, we want to train the model constantly, such that new data and model benchmarks are as close as possible to reality.

When we use models for process optimization, we might need yet another feedback to know what we do with machine learning model outcomes. Going through this thought process, we realize that the application of machine models is more than a simple pipeline problem. It eventually becomes a process automation – orchestration problem.

Serveless API with Firebase
Michael Dowden | Andromeda Galactic Solutions
Are you currently using microservices and are looking to move to serverless? Are you wanting to build a new serverless application, but you need to provide a public API? Both of these needs can be addressed with Firebase Cloud Functions. This talk will show how to build a variety of API endpoints with Cloud Functions, including some operational details such as resource allocation, available instances, run time, and middleware integration. You’ll walk away with the basic knowledge to implement Cloud Functions and the confidence that it can work for your use case.

Links & Literature
Big changes for Software as a Service

“We are in the early stages of a huge shift away from building SaaS product capabilities in-house”

We spoke with Sagi Rodin from Frontegg about the challenges that SaaS companies face with ever increasing security threats and rising expectations from customers for greater control, freedom and independence. Sagi also talks about how utilizing easily integrated, full stack product capabilities can help solve this by allowing startups to keep focused on their core product, and launch faster with a more mature offering.

Interview with Sagi Rodin

Serverless Architecture Conference: What are some of the big changes that you have seen over the past few years in how SaaS is being built?

Sagi Rodin: In recent years we have seen SaaS companies move towards integrating third party, full end-to-end capabilities, into their products. In the past, SaaS companies were hesitant to use externally developed product capabilities in customer-facing parts of their applications – with the notable exception of Stripe or similar solutions for payments. They were happy to use third party infrastructure solutions or open source UI component libraries. But when it came to a full product capability with customer facing parts, they inevitably opted to develop it in-house.

Serverless Architecture Conference: Why were the decision makers in SaaS companies hesitant to incorporate externally built product capabilities?

Sagi Rodin: I think that a lot of this comes down to inertia; a kind of chicken and egg problem. Companies didn’t consider third party solutions an option, so as a result, great third party solutions weren’t developed, so there were no good solutions for companies to consider – it was a never-ending cycle. There was a status quo where the default was to develop customer-facing SaaS capabilities in-house. Decision makers were stuck in this mindset and rejected suggestions to incorporate externally built capabilities. The common outlook was: “why pay for something we can do ourselves?” or “That’s what we hired all these developers for.”

Plus, there were additional barriers to overcome as well. There was a fear of the security implications of shipping third party code to customers, and the difficulty in integration being caused by the chaotic nature of the web development ecosystem — which suffers from a lack of standardization. These barriers prevented things from becoming unstuck, and even as SaaS companies
began to look to third party solutions for things such as marketing sites, infrastructure, and internal project management tools, the adoption of these solutions lagged behind for in-app, customer facing capabilities.

**Serverless Architecture Conference: What do you see as the key things that changed?**

Sagi Rodin: Stripe actually played an important role in breaking this cycle. They tackled payments, which is a capability so complex, no company would consider building it on their own. Stripe capitalized on the move towards self-service SaaS [1] and the demand for in-app payment solutions that came as a result and helped accelerate the move to self-service by providing a payments solution that was orders of magnitude simpler to integrate than previous options. Stripe made the brilliant decision to not just supply an API for payments but to offer an end-to-end solution – that includes customer facing payment flows – with just a few lines of code. The simplicity of their solution, combined with the complexity of implementing payments made them the obvious answer for many SaaS companies attempting to keep up with the growing demand for in-app payments driven by the movement towards self-service and Product-led growth.

Following Stripe’s lead, a number of other companies started offering full end-to-end product capabilities, including user facing parts, with very simple integration. The status quo has shifted and it’s now common for products to have a number of third party user-facing integrations, for not only payments, but also login flows and user onboarding, among others.

**Serverless Architecture Conference: What about the security implications of including third-party code in the application that you’re shipping?**

Sagi Rodin: Interestingly enough, security has actually shifted from originally being one of the main concerns, to today serving as a major benefit. Security is now seen as a constantly escalating arms race, with the attacks becoming more and more sophisticated with time. It is increasingly difficult for developers to stay at the cutting edge of the latest security measures and best practices. This is especially true when it comes to features with significant security aspects — features like Authentication, User Management, Roles and Permissions and API Keys. Today, developing these features in-house can become a liability. This is because properly implementing these capabilities is so complex and has so many nuances, it requires the knowledge of a security expert — something not every development team has. So, if instead you can integrate externally developed capabilities, after confirming that the supplier has the relevant security credentials (and takes it seriously), you can sleep easier at night knowing top experts who live and breathe security are keeping your customers safe.

But security alone is not enough. The most recent trend is that security is no longer just an infrastructure challenge. Within the product-led growth (PLG) movement, end-users would like to control the security narrative of their workspaces within any product they are utilizing. As a result, SaaS companies need to provide full self-served control of any security policy setting to their customers, within their products. This is what we call “inversion of control” of next-generation security within applications. Just imagine that an end user wants to configure their Active Directory within a SaaS app and in order to do so they need to open a ticket, talk to support, exchange metadata configuration files, and validate that it works while talking to a Customer Success rep over the phone. It just doesn’t work like this anymore. Customers want independence and full control over their workspace policies, MFA enforcement, SSO setup, granular and custom roles and permissions and much more.

This means that it’s no longer enough to just provide strong security infrastructure as a service, but you have to also provide it embeddable to your customer’s user interfaces. This of course greatly increases the complexity of implementing these capabilities. For example, a customer of ours needed to add SSO to their SaaS product and make it fully configurable and self-served for their end users. Their R&D team’s best estimate on the development effort was 7 to 8 months, but they had deals in the pipeline that were dependent on this. By integrating Frontegg’s SSO solution [2] they were able to turn this around in a matter of weeks and close deals that would otherwise have been lost.

“Although the web development ecosystem is still a bit messy, integration of third-party capabilities has become much more natural with the move away from monolithic apps towards microservices and micro-frontends.”

**Serverless Architecture Conference: What has changed on the integration front?**

Sagi Rodin: Although the web development ecosystem is still a bit messy, integration of third-party capabilities has become much more natural with the move away from monolithic apps towards microservices and micro-frontends. At the same time there has been a convergence on a small number of frontend libraries and frameworks, like Angular, React and Vue.js. These are all designed to be component-based and composable, which really lends itself to creating easy to integrate APIs. It has also opened the door for third party capabilities that have a large number of integration points and need to be finely woven into the application code, as opposed to solutions like Stripe that have a relatively small surface area for integration.
Serverless Architecture Conference: Where are things heading in the future?

Sagi Rodin: I think we are in the early stages of a huge shift away from building SaaS product capabilities in-house by default. Instead, we’ll look first for existing plug-and-play solutions and only develop in-house if we reach the conclusion that there’s nothing suitable out there. The result will be that most products will have significant portions that are third party integrations, while some products will be almost entirely composed from plug-and-play capabilities. Developing product capabilities from scratch will be reserved for cases when the capability is truly unique, innovative and core to the product.

We are already seeing the adoption of these kinds of solutions for in-app analytics (for example Looker), data onboarding (FlatFile) and search (Algolia). This is just the beginning though, and we will see this process accelerate over the coming years.

As customers will demand more independence and control over various aspects within their workspaces, SaaS vendors will have to deliver. Leveraging third-party product capabilities as a service provides an efficient manner to do so.

But in order for these capabilities to become widely used two things need to be tackled. Firstly, they need to provide an amazing developer experience. This goes beyond just a clean, intuitive API. It requires clear, up-to-date and comprehensive documentation, and great support for developers generally, putting them front and center.

Secondly, they need to offer a great user experience. SaaS companies cannot – and should not – compromise on user experience. The user is king and every “security features” like login need to be designed with a seamless experience for the user in mind.

Serverless Architecture Conference: What are the benefits of leveraging third-party capabilities? What is wrong with the logic of “why pay for something we can do ourselves”?

Sagi Rodin: I think the major flaw in this kind of thinking is a failure to properly account for the total cost of ownership (TCO). When you think about the cost of developing a product capability, the initial development costs are only the beginning. There are continual maintenance costs that arise from fixing bugs, adapting to changing product specifications and adding new capabilities.

Not to mention that the code for any given feature has a kind of half-life – some period of time after which the code will become legacy code and will need to be entirely rewritten. This is because the tech is constantly evolving – frameworks, libraries and best practices are always changing – and because shortcuts or assumptions that were made in the initial development may not hold up over time. The stubborn reality is that at some point you’ll ask an engineer to make a small fix or change to a feature that was hacked together years ago over the course of a few weeks, and they’ll come back saying that they can’t keep on putting patches on patches and they need to do a full rewrite. When integrating a third-party capability, the maintenance is done by the third-party behind the scenes.

Another consideration is the opportunity cost. We don’t have limitless resources and when you build a generic and non-differentiating capability yourself you are forgoing on building another capability that could make your product stand out from the competitors.

Finally there’s focus. Startups rise and fall based on how successful they are at remaining laser focused on their core mission. Offloading generic development work by using pre-existing solutions allows the development team to stay fully focused on the core product that makes your offering stand out.

Especially when it comes to capabilities for self-service and Product led-growth, there’s a tremendous opportunity to integrate ready made solutions. This enables teams to go to market fast with a fully mature product that can scale up quickly, without needing to sacrifice on the core product capabilities or on the self-service features that drive growth.

Links & Literature

[1] https://frontegg.com/sso

---

Serverless Integration with Zapier

Michael Dowden | Andromeda Galactic Solutions

Integration between 3rd party applications and APIs has long been one of the most challenging and time-consuming parts of any application development effort. But with the magic of cloud automation, we can connect applications in just minutes instead of days. In this talk, you’ll learn about responding to cloud events such as HTTPS endpoints and Tweets, connecting multiple APIs such as MailChimp and HubSpot, how @TechDailyCFP was built using cloud tools without a single line of code, and the benefits and drawbacks of cloud automation tools. Examples will be on the Zapier platform, but similar principles apply to Integromat, Azuqua, Automate.io, and others.
Collect data, analyze, and respond

Strategies for the Cloud Threat Hunter

Threat hunting, like all cloud security, is rendered additionally difficult by high volumes of transactions and the high velocity of deployments. Security professionals face constantly evolving landscapes as their company infrastructures change and cloud providers offer new tools. While standard and cloud threat modeling share basic methodologies and joint purposes, there are meaningful differences.

By Asaf Shahar

No matter how strong its security defenses are, your cloud infrastructure is not an impenetrable fortress. There will always be issues such as vulnerabilities or misconfigurations in the cloud, and it is vitally important to be prepared for that. While maximizing defenses to prevent attacks from succeeding is crucial, cybersecurity teams must also take the offensive and build an arsenal of tools and tactics to hunt threats.

The Unique Challenges of Cloud Threat Hunting

Threat hunting, like all cloud security, is rendered additionally difficult by high volumes of transactions and the high velocity of deployments. Security professionals face constantly evolving landscapes as their company infrastructures change and cloud providers offer new tools. While standard and cloud threat modeling share basic methodologies and joint purposes, there are meaningful differences.

Cloud threat hunting is a never-ending story. In a perpetual loop, we find threats and fix them, then attackers pivot, and the cycle repeats. Unfortunately, both the quantity of these loops and the speed at which they change increase exponentially in the cloud. In addition, cybersecurity professionals often must track multiple cloud vendors and multiple assets, many of which are ephemeral.

Reduce the Detection Gap to Limit Dwell Time

One growing strategy cyber attackers deploy is privilege escalation. Once they’ve accessed a cloud environment, attackers move laterally to access sensitive assets. It’s vital to detect threats swiftly in order to reduce the time they can dwell within your environment and cause damage.

For example, privilege escalation begins by an attacker accessing a user account with low privileges. Attackers then use techniques to remain undetected, such as inserting a payload into user data to avoid creating a new key pair to launch an EC2 instance, ultimately gaining access to credentials.

Unfortunately, most organizations are failing to limit dwell time. In fact, it takes an average of 111 days for an organization to realize they’ve been infiltrated. This is why it is important to rely on techniques, like those outlined in the MITRE ATT&K® framework, to detect and thwart attacks.

Identify: First Step to Cloud Threat Hunting

Threat hunting requires making observations, collecting information, creating hypotheses, analyzing data, and investigating to prove or disprove hypotheses. To collect data, analyze, and respond, the right tools and processes must be in place.

1. Collect Data – Hunt for Indicators of Compromise (IOCs)

Threat hunting starts with collecting quality data from various sources, including logs, servers, network devices, firewalls, databases, and endpoints. Spotting malicious activity here can enable you to lessen the impact of any breaches and prevent mega-breaches.

Doing so requires having a solid understanding of both normal use on your network and how attacks are perpetrated. Strong IOCs include a malware signature
alert on your network and ransomware executables on your file system, picked up by your intrusion detection system (IDS) or anti-virus. Examples of weak IOCs are repeated failed user login attempts and login times which align with typical use.

Set IDS alerts only on strong IOCs to help avoid alert fatigue. When chained together, weak IOCs can build a strong indication of compromise. Human insight is needed to do the work of chaining indicators together.

2. Analyze
When investigating suspicious activity in your cloud infrastructure, try to break down threats into the following detectable components:

- How the threat entered their environment;
- What it affected;
- How it currently impacts their organization wherever it persists.

Next, analyze the data gathered to search for patterns and detect IOCs. Again, identifying anomalies such as abnormal account activity requires having an established baseline of normal use.

3. Respond
After progressing through an iterative pattern of collecting and analyzing data, you must draw conclusions and respond accordingly.

Visibility + Tools
Effective threat hunting is dependent on the quality of your tools and the visibility you have into your environment. Unfortunately, visibility in the cloud is a challenge for most organizations, especially when it comes to what data is within their cloud applications. If you don’t know where your high value data is, you can’t protect it. Inability to monitor data in transit to and from cloud applications is a common blind spot.

Threat modeling is the act of identifying potential threats and then modeling avenues of attack. This exercise enables you to prioritize and mitigate risks. When threat modeling, consider questions such as what do you want to protect, what are the consequences if you fail, and how much trouble are you willing to go through in order to prevent those consequences.

Recently, the Cloud Security Alliance (CSA) released a Cloud Threat Modeling guide [1] which features a framework that organizations can use to create their own cloud threat model. Similarly, Microsoft outlines attack trees, “a way of identifying and documenting the potential attacks on your system in a structured and hierarchical manner.”

Be sure to conduct tests by simulating a variety of threats in the cloud, such as mimicking cross-tenant attacks, and study real-world attacks against cloud infrastructure. For those seeking to learn, there are many to choose from. Furthermore, threat hunters should produce attack patterns and “misuse cases,” and map out the processes of attack and defense or countermeasures sequences.

Threat Hunting at Scale
For the vast majority of organizations, collecting and analyzing data cannot be done manually. Conducting ongoing intrusion detection and monitoring manually requires an exorbitant amount of resources and manpower, and is simply impossible in modern cloud environments. Automation is necessary, and threat hunting must be part of a unified approach. Be careful, however, as an excess of tools can convolute threat hunting. SIEM, for example, doesn’t extract information in a way that’s easy to understand.

It’s important also to set real time alerts that correspond to different attack techniques outlined in the MITRE ATT&CK framework. Alerts must be prioritized based on risk, and as a breach persists and escalates, continue to alert users to IOCs as they occur.

In order to stay ahead of the attacker, you need to leverage these methodologies across your cloud and on-premise environments. The more you can consolidate and integrate your tooling, the greater visibility and operational efficiency you will have. Along with the use of automation and alerts in context will allow you to scale and hunt for the next critical threats jeopardizing your cloud.

Asaf Shahar is the Sr. Product Manager, Cloud Native Security at Check Point Software Technologies. Asaf is a senior technology and product executive, with a proven track record of over 18 years of defining and delivering large scale complex products, architectures and technologies. Vast experience in security, cloud and networking.

Implement a Serverless Data Analytics strategy with Microsoft Azure
Roberto Freato | Witailer

After a decade of innovation around big data, distributed filesystems, data movement tools, and scalable data warehouses, I want to share my thoughts on how Microsoft Azure tries to wrap all this stuff with a few tools, like Synapse, DataLake Storage, and more. As a result, we will implement a Data Analytics strategy in a serverless mode, on a consumption-based model.

Links & Literature
Making sense of cloud-native complexities

“Lumigo is purpose-built for cloud-native”

Erez Berkner, the co-founder and CEO of Lumigo, spoke with JAXenter about the cloud observability space, elaborating on what it’s trying to solve and how it’s different from monitoring. He explains in detail how Lumigo helps make sense of the ever-growing complexity of cloud-native applications and allows developers to find the root cause of an issue in highly-distributed systems. Erez explains Lumigo’s decision to expand to Kubernetes and hybrid apps and reveals the company’s plans for further down the road.

Interview with Erez Berkner

Serverless Architecture Conference: How is observability different from cloud monitoring and testing?
Erez Berkner: Monitoring enables teams to keep an eye on the state of their systems and understand them. It involves collecting predefined metrics or logs. People usually use monitoring to keep track of a system’s health. They do that by collecting error logs and system metrics, and then using those to alert about issues.

Observability, on the other hand, lets teams troubleshoot and debug their system. It spots patterns that

Fig. 1: Many dependencies across internal and 3rd party services

serverless-architecture.io  @ServerlessCon  # ServerlessCon
aren’t defined upfront. With observability, we gather insights and act on them.

Serverless Architecture Conference: What are the main pain points cloud observability is trying to solve?
Erez Berkner: Cloud systems, and particularly those built with serverless and microservices architectures, are typically highly distributed with many dependencies across internal and 3rd-party services. (Fig. 1) This creates a particularly painful challenge when trying to get to the root cause of an issue, such as a bug or performance problem.

Serverless Architecture Conference: How does Lumigo solve these pain points?
Erez Berkner: At Lumigo, we use an approach known as automated distributed tracing, which tracks a service request across all of the services – whether internal or external – that are required to complete a transaction. In addition, it gathers in one place all of the relevant information that would help a developer identify the root cause of an issue and resolve it. Our platform correlates logs and creates a virtual stack trace of your distributed environment (Fig. 2).

Serverless Architecture Conference: How do you correlate millions of log lines, traces and metrics across distributed services? Can you explain how this works under the hood?
Erez Berkner: Sure. Lumigo’s correlation engine uses innovative algorithms that were developed specifically for cloud services. The engine uses data observed by the Lumigo tracer (a code library) to deterministically identify a request and correlate the logs, inputs, env variables and traces across distributed services. It’s a no-code, no-deployment concept that is able to correlate synchronous managed services (e.g., Lambda or Stripe) as well as asynchronous managed services (such as dynamoDB).

Serverless Architecture Conference: Cloud-native applications are getting more complex every day. What future do you see for cloud-native applications?
Erez Berkner: The good news is that new cloud-native services are released by cloud providers all the time, which makes developers’ lives easier in certain respects, and lets applications run more efficiently. But there is a flip-side which is increased complexity. We definitely see this trend continuing and it’s really the reason the Lumigo correlation engine is required to help developers identify the root cause of an issue.
go platform exists: to cut through all the complexity and give developers an easy way to understand what’s going on with their application.

Serverless Architecture Conference: What is the demand for your platform? What kind of response are you getting from developers?
Erez Berkner: The demand for our platform is directly tied to the maturity level of cloud-native usage — meaning in-production workloads. So as you can imagine, it’s quite high right now... The response has been tremendous. On our Slack, we have an internal channel we called #customer-compliments, which gets multiple posts every day, with things like “We connected Lumigo 20 minutes ago and immediately found problems we weren’t even aware that we had and are now working on implementing fixes!” It’s very satisfying and gets us excited every time.

Serverless Architecture Conference: How is Lumigo unique?
Erez Berkner: I’d say the main way in which we’re unique is that we are completely obsessed with cloud-native. We are not a legacy monitoring tool that’s trying to create a patchwork of solutions to address these new environments. Lumigo is purpose-built for cloud-native. We are literally 100% cloud-native ourselves and our platform was developed by cloud-native developers for cloud-native developers. The way it manifests itself in the product is by having the right metric or debugging info available for the developer at the right time, as well as having the right cloud-native out-of-the-box metrics (for example, cold starts or service latencies) and these features really speak to cloud-native developers and make them say “these guys get it”.

Serverless Architecture Conference: Lumigo recently raised $29M to expand the platform to Kubernetes and hybrid apps. Why did you choose to focus your efforts in this direction?
Erez Berkner: So first, I want to make sure your readers know that we are still very much focused on serverless. The important word in your question is “expand”. The reason we’re doing this is because our customers are telling us that in the real world, their applications are usually built on a mix of serverless and non-serverless environments. They use serverless and managed services but also containers, Kubernetes and even VMs. And our goal is to give them a holistic view of what’s happening in their applications and be able to easily resolve any issues.

Serverless Architecture Conference: What can we expect from Lumigo in the future?
Erez Berkner: One important direction is exactly what we mentioned above: making sure we give developers a complete end-to-end picture, regardless of technology or specific cloud services. We want them to be able to use Lumigo on any modern cloud technology. And as the cloud-native ecosystem evolves and grows, so will Lumigo.

The second area of focus goes back to how we opened this discussion with the difference between monitoring and observability. We want to give developers the best tools to find and resolve issues and we’ve added many capabilities for that purpose lately, such as identifying rogue deployments or seeing a “Live tail” of the application logs. Many more are on the way.
Find the right mix of value to the business and value to IT

A Cloud Data and Analytics Platform Has Major Benefits, But That Doesn’t Mean it’s Easy to Manage

As the phrase “move to the cloud” becomes ever-present, it’s critical to understand that just like any data engagement, a cloud data and analytics program needs to be treated with the same attention to strategy, process and testing. While we’d like to believe that it’s as simple as spinning-up a new environment and moving data, doing the project correctly (and planning for future growth and success) is actually far more involved.

By Dave Taddei

There are three things to consider when moving your data and analytics to the Cloud. In an increasingly digital world, we’re seeing a common trend with our clients: the misperception that cloud equals easy. As the phrase “move to the cloud” becomes ever-present, it’s critical to understand that just like any data engagement, a cloud data and analytics program needs to be treated with the same attention to strategy, process and testing. While we’d like to believe that it’s as simple as spinning-up a new environment and moving data, doing the project correctly (and planning for future growth and success) is actually far more involved.

Understand that data management and data and analytics patterns have changed in the cloud
The cloud has brought about the separation of storage and computing. That fact alone means that the old modes of operating have fundamentally changed. As an example, the shift from extract-load-transform (ETL) to an extract-load-transform (ELT) pattern. With the cloud you have instantaneous scalability, so you don’t want to bog-down the initial load of that data. The cloud reduces your costs and load times and allows you to centralize both data and logic – but you have to understand how to do it. Knowing the rules of this “new game” are key to building your architecture for success, and operating in this space requires data strategists who know the cloud. When it comes to this skill set, not all data teams are the same.

**Your cloud data and analytics platform still needs to connect to your broader ecosystem**

Your cloud data platform is not going to be in a vacuum – you’re still going to have a number of systems that exist on-prem, and your cloud data platform needs to understand that the business and operations still need to function. As you take steps to move your data and analytics processes into the cloud, you have to understand what the current systems do and how to swap them out with limited to no interruption to the business itself. These internal systems need to be updated to account for the introduction of a cloud-based data analytics strategy, and the team needs to be trained on how to incorporate this new infrastructure into their existing processes. There is going to need to be change, and that change needs to be planned for, documented, and managed.

**Balance process and speed when delivering data and analytics to the business**

Can a data and analytics move to the cloud be fast? Absolutely. We can move quickly and get an initial analytics output, but if the initial steps weren’t well-thought-out, you’re not building something to last. Typically what we see is that a rapidly deployed cloud solution tends to run into the same challenges: the business likes what they see, and they want more data and more use cases. They’re getting value, increasing revenue, increasing efficiency, and they want more of it – at the same speed. But they can’t scale fast enough because the foundation isn’t built for that speed of expansion.

Another key drawback to speed at all costs is that oftentimes the team who is tasked with maintaining that pace is learning as they go, and as a result, mistakes are made:

- Security models get missed or aren’t defined: as you scale, privacy concerns become more of an issue as you bring in more data sources. A good security model needs to be built from the beginning. You’re giving people access to data from across many systems, and it’s much more critical that you manage the security of your platform.
- No clearly-defined data ops process: how do you control your versions, source code management, and deploy and merge changes from staging into production? If not defined, those basic systems all have to be rebuilt later, and it ends up being a far more difficult (and slow) process than if they were defined from the start.
- One person owns it: we see this all the time. A singular person is tasked with owning the cloud data and analytics architecture, and they have the keys to the kingdom. No one knows how to do what they do because nothing is systematized. When they go on vacation and something breaks, it’s catastrophic.

Remember, data and analytics in the Cloud is ‘new,’ but the approach isn’t. It’s critical to find the right mix of value to the business and value to IT. We can’t forget the lessons of the past and best practices of on-prem data processes. This is simply about translating those processes and evolving them to work in a new environment.

---

**Designing business processes using the Serverless architecture**

Michał Górski | Northmill Bank

We have very different business processes. Sometimes they are held once a month. Other times the traffic is unpredictable over time. The perfect solution for such cases is Serverless architecture. Thanks to this approach, we can optimize the cost of the process, as well as obtain high availability even when the traffic is unpredictable. In this session, I will show a few examples of how to design business processes using the Serverless approach. We will focus on serverless and managed services and how to connect Lambda, SQS, EventBridge, S3 Bucket, DynamoDB, etc. to build highly scalable and cost-optimized applications.

**Dave Taddei is SVP, Data Analytics North American Practice Lead, at AllCloud.**
Moving towards no-code/low-code

No-code is SaaS 2.0

In 2022, we’re going to see a move toward no-code/low-code (NC/LC) startups and offerings, which will address portal fatigue as well as offer a better value proposition. NC/LC offerings are the clear next step for business functions as their flexibility, best practice templates, and ease of integration with current and future solutions make them compare quite favorably to the SaaS solutions that have begun to encumber all of our work machines.

By Jag Lamba

To say that SaaS was a game changer would be under-selling the monumental impact the model had. Whole industries changed so fast it seemed like it happened overnight, and software implementation transformed from a headache-inducing, business-halting process into the turning of a key.

Until very recently, this model proliferated with great success, and for good reason. There were so many business functions that still relied on spreadsheets, the digital equivalent of cave paintings (remember, Excel was first released in 1987 – 35 years ago, which is practically eons when we’re talking technology), and it was simple to take those functions online, add workflow features, reporting and analytic functions, and voilà! A business with recurring and reliable revenue.

But over recent years, these first-generation SaaS solutions (SaaS 1.0) solutions have flourished to the point that it’s not unusual for an average business user to need to log into several systems to accomplish a single task or goal. As every function went online, the various systems and logins and processes out there proliferated to the point where people started suffering “portal fatigue,” frustration or energy drain due to the process of managing so many different login credentials, workflow and reporting systems, and all the additional details that come with a new system – not to mention trying to get them to work together, which wasn’t always easy.

In 2022, we’re going to see a move toward no-code/low-code (NC/LC) startups and offerings, which will...
address portal fatigue as well as offer a better value proposition. NC/LC offerings are the clear next step for business functions, as their flexibility, best practice templates, and ease of integration with current and future solutions make them compare quite favorably to the SaaS solutions that have begun to encumber all of our work machines.

NC/LC solutions use dynamic data models, workflows, and in-app reporting systems to provide flexibility that current SaaS point solutions often can’t match. While the first wave of SaaS solutions (hence the 1.0) was typified by a sense of “the work you’re used to—but online,” these new NC/LC solutions can be adapted by the end users to better fit the kind of work they need to do. Those with zero programming skills can use the tools in the solution to adapt it in a way that they’d need to bring on tech teams to do today. That’s the “no-code” part of the name; “low-code” solutions might still need some level of programming ability to make changes to better adapt the solution to their use case, but the barrier is still much lower than it is with higher-code-requirement applications.

No-code software in particular has exciting applications for future business environments. Users will be able to create both web and mobile applications and digitize complex business process with a simple drag-and-drop; no code-writing required. And since these applications aren’t usually tied to a specific use case, but a broad solution, it’s easier for software providers to offer starter templates that clients can use to make the no-code solution meet their specific needs.

The next wave of NC/LC solutions we’re likely to see in 2022 also bring benefits in extensibility, integration, and more. Extensibility is the ability to add adjacent use-cases; spending a year with a point solution only to learn that you have to either abandon it in favor of something more comprehensive or find another solution and integrate it with your current software stack, is not a scenario that anyone should find themselves, but many do today. Along similar lines, many NC/LC tools come with powerful integration engines that will create a unified platform for users that integrates with enterprise ecosystems already in place. And while SaaS made product updates easier to deploy, NC/LC allows for easier continuous updates to keep things humming all the time. And the ability to utilize multiple best practice options – not just what the vendor suggests at the outset of a SaaS deployment.

2022 will bring an acceleration in both availability and adoption of these no-code/low-code solutions. It’s the beginning of the change that will come to a head in the next half decade. In two to three years, we’ll see many of these platforms integrating with and orchestrating existing SaaS point solutions, relegating those 1.0 solutions to backend databases for business users. Then in a few more years – three to five years from now – I expect NC/LC solutions to overtake and fully replace SaaS point solutions. The benefits and flexibility of these solutions are too attractive for businesses to pass up, and since they directly address the downsides of the SaaS 1.0 solutions we experience today, that adoption will be rapid once it comes. There are already some NC/LC solutions out there if you know where to look, and it’s only a matter of time before they’re the de-facto standard for businesses across industries.
Interview with Matt Biilmann

Serverless Architecture Conference: You’ve been talking about Jamstack for a long time now, what makes it such a strong stack for the serverless model?

Matt Biilmann: From the onset, Jamstack represented an approach to web development where the traditional concept of “my server” went away and where the stack moved up a level. When serverless functions started emerging, they instantly felt like a really compelling part of the Jamstack, since they embodied the same shift in the level of abstraction, from working with servers, Linux distributions, system packages, etc, to just writing code and then having it run.

Jamstack allows developers to decouple the web presentation layer from the backend logic, and often this logic becomes a great fit for a serverless approach.

The vision of Jamstack has always been to pre-render whatever you can and then pull any dynamic data that you need. One of the core tenets of Jamstack is the use of reusable APIs, which can just as easily be a serverless function written as an API endpoint. That means a frontend developer can now create more of the experience and easily call on backend services without coordinating across DevOps and backend teams or provisioning servers and APIs. Most importantly, Jamstack platforms like Netlify [1] allow you to bring the complete development workflow together in one place—where the entire app, including serverless capabilities, can be managed and deployed from a single Git repo.

Jamstack and serverless share the idea that developers should be able to focus on building experiences without having to worry about operations and infrastructure provisioning. Both start with the emphasis on developer experience and developer productivity which make the two natural companions. When managed from a single platform like Netlify, it means that you can apply a frontend workflow to everything, using advanced features like instant rollbacks, easily shareable previews, split testing, and more across both the front-end and serverless backend parts of your app.

Serverless Architecture Conference: Is the use of Jamstack still becoming more widespread as we see...
more cloud transformations happening out of necessity because of the pandemic?
Matt Biilmann: Yes – we’ve seen a big acceleration in digital transformation projects and just more attention to delivering a dynamic and performant web experience. The pandemic moved face-to-face interactions online faster than anyone could have predicted, making Jamstack an even more compelling case for development teams.

Jamstack takes a huge leap forward in speed. There’s almost no latency in the web experience because the heavy lifting happens when the app deploys – when it’s actually compiled and distributed to the network edge. From the users’ perspective, the web app loads instantly because it’s served right from the network edge, without having to render everything on the fly each time, which is the case in a non-Jamstack environment. And of course, faster sites and web apps mean a more competitive experience, improved conversion for e-commerce, higher SEO performance, and so on.

Jamstack is also a far more secure approach to the web. Because you’re removing the dedicated web server as a vector for attack, the surface area for threats is dramatically reduced, meaning that a developer can focus on iterating the web experience and not chasing the latest security issue.

Scaling is also much easier and less expensive with Jamstack which is incredibly important as people move online for more things. In this case, the web frontend lives right at the network edge across many global locations and can elastically scale to accommodate traffic spikes. No capacity planning needed.

Finally, it frees developers to work even faster and use more modern tooling because now frontend development is independent of the backend infrastructure.

In the context of the pandemic, headless commerce is one good example of a Jamstack approach to address the skyrocketing demand for online shopping. “Headless” refers to the approach of decoupling the data and underlying functionality from the frontend UI. It’s really freed up development teams to move faster and ship more engaging and personalized shopping experiences. Experiences that drive traffic and convert at a much higher rate.

Serverless Architecture Conference: What are some challenges of the serverless model?
Matt Biilmann: We still see development teams spending a lot of time implementing backends and optimizing performance of the experience to end users.

While some of the implementation challenges have been solved if you’re just looking at deploying a serverless function, things get pretty tricky as you look at delivering serverless across the entire web experience. The app might call on a serverless function for one piece of data, call on a longer running function to accomplish something asynchronously, trigger a custom behavior at the network edge such as localizing content, and use compute resources to compile an update to the static content when a change is made from the CMS. All of this can call on serverless compute resources. What teams want is a front-end approach to orchestrating all of these services.

The second challenge has to do with performance. If developers are looking at server side rendering alone to deliver a web experience, it’s typically too slow and can’t compete with the speed and efficiency of pre-rendering. That’s why development teams are turning to Jamstack as a way to get the best of both worlds.

Serverless Architecture Conference: And how might those challenges be overcome?
Matt Biilmann: To overcome implementation challenges, Jamstack platforms offer a simple frontend workflow for developing and accessing cloud-scale backend services. Serverless functions, long-running asynchronous functions, edge compute logic, and pre-compute resources to compile the app – are all managed and deployed alongside the UI code from a single repo.

To overcome the challenge of performance, teams are moving to Jamstack, which allows for a more hybrid approach, taking the best parts of serverless and combining it with lightning-fast pre-rendered content. You effectively remove the performance tradeoff of serverless because static elements of the website are pre-rendered, and then a serverless backend is delivered either via APIs or right from the network edge in response to request/response logic.

Serverless Architecture Conference: What’s the future of serverless?
Matt Biilmann: When serverless started, it was about making the lives of backend developers easier. As it’s progressing, we’re seeing more front-end focused teams using serverless to build APIs and access data that wasn’t easily accessible. Serverless is going mainstream. The new assumption is that – if you can build business logic in code, then you should be able to build an API that works as a serverless function. Frontend teams don’t just build UIs anymore – they can start to help themselves with API-accessed backends that cater to their requirements.

Mathias Biilmann is CEO of Netlify, a company he co-founded in 2014 and today is one of the fastest growing web development platforms. He has been building developer tools, content management systems and web infrastructure for more than 30 years and is recognized for coinining the term “Jamstack.” An active participant in open source, Matt has contributed to well-known projects in more than a dozen languages at all layers of the stack. Matt grew up in Denmark, where he trained as a musician and music journalist.

Links & Literature