THE DZONE GUIDE TO

BUILDING AND DEPLOYING APPLICATIONS ON THE CLOUD

VOLUME III

BROUGHT TO YOU IN PARTNERSHIP WITH

New Relic.

NGINX

Iron.io

mx mendix

CloudBees
The Enterprise Jenkins Company

IBM

DEAR READER,

The cloud isn't cool anymore—everyone uses it all the time. It's just Von Neumann machines whose physical implementations you don't (usually) particularly have to think about. In that sense, Larry Ellison was right: "All the cloud is, is computers on a network."

But hang on a minute. Ten years ago, could you serve a million simultaneous users without ever talking to an IT admin?

If not, then *something* about how we move our code to production has changed since "cloud computing" started trending on Google.

Overhyped or not, the cloud has deeply changed how we build and run software—and not just because IaaSes make VMs trivial to spin up and PaaSes make environments easy to set up. As a user you know what's changed, and you understand the concept "as a service" (well, ever since you started running *nix); and, thank goodness, you don't really have to worry about the physical details that make those services run.

But how can you take advantage of the granularity, elasticity, and pre-baked-ness of modern cloud services? And, on the flip side, as your applications expand outside blade-boxes and even beyond individual data centers, how do you design (and troubleshoot) applications that treat network links like second-generation buses, that won't hang if a service a continent (and an SLA) away just isn't working, and that store sensitive data on machines whose geolocation and hardware configuration you don't even know?

We've done our best to pack concrete, immediately implementable solutions to these and related problems into our ***2016 Guide to Developing and Deploying Applications on the Cloud***. We hope our original research, articles and checklists, and pull-out poster visualizing the 12-factor methodology will help you design and run fast, reliable, easy-to-update cloud-native applications on any infrastructure, in any vertical, for any use case.

Browse, absorb, test, critique, and let us know what you think.

BY JOHN ESPOSITO
EDITOR-IN-CHIEF, DZONE RESEARCH@DZONE.COM

TABLE OF CONTENTS

EDITORIAL
JOHN ESPOSITO
RESEARCH@DZONE.COM
EDITOR-IN-CHIEF

CAITLIN CANDELMO
PUBLICATIONS MANAGER

ANDRE POWELL
EDITORIAL OPERATIONS MANAGER

G. RYAN SPAIN
ASSOCIATE EDITOR

MATT WERNER
ASSOCIATE EDITOR

MICHAEL THARRINGTON
ASSOCIATE EDITOR

TOM SMITH
RESEARCH ANALYST

BUSINESS
RICK ROSS
CEO

MATT SCHMIDT
PRESIDENT & CTO

JESSE DAVIS
EVP & COO

KELLET ATKINSON
VP OF MARKETING

MATT O'BRIAN
SALES@DZONE.COM
DIRECTOR OF BUSINESS DEVELOPMENT

ALEX CRAFTS
DIRECTOR OF MAJOR ACCOUNTS

CHRIS SMITH
PRODUCTION ADVISOR

JIM HOWARD
SR ACCOUNT EXECUTIVE

CHRIS BRUMFIELD
ACCOUNT MANAGER

ART
ASHLEY SLATE
DESIGN DIRECTOR

SPECIAL THANKS
to our topic experts, Zone Leaders, trusted DZone Most Valuable Bloggers, and dedicated users for all their help and feedback in making this report a great success.

WANT YOUR SOLUTION TO BE FEATURED IN COMING GUIDES?
Please contact research@dzone.com for submission information.

LIKE TO CONTRIBUTE CONTENT TO COMING GUIDES?
Please contact research@dzone.com for consideration.

INTERESTED IN BECOMING A DZONE RESEARCH PARTNER?
Please contact sales@dzone.com for information.

EXECUTIVE SUMMARY

Cloud technologies have been gaining traction for some time now. Increases in connectivity throughout the computing world with the creation of more and more connected devices, including mobile and IoT technologies, as well as more and more connected applications on those devices, means cloud computing adoption is ever-increasing. Expectations regarding an application's availability are high, and solutions continue to emerge to increase availability and make scaling applications easier when a user load gets too big. New patterns, platforms, services, and software are pushing applications and data to the cloud. To help you sort through these, we've assembled information and advice in this guide that can assist you in choosing, setting up, and maintaining cloud platforms, and using the cloud to optimize your application. **This guide includes:**

- Articles from six industry experts
- A directory of cloud tools to consider when building a cloud-based application
- A checklist for making sure your cloud-based application is secure
- Analysis of trends from a survey of over 700 IT professionals
- C-level perspective on cloud computing based on from interviews with 28 executives

DEVELOPERS ARE ADOPTING CONTAINERS QUICKLY

DATA 66% of survey respondents say they are evaluating or using container technologies in their organization right now. 92% of those respondents say that containers are being used or evaluated in development. 35% of respondents whose organizations are currently using containers release their applications on-demand (several times a day); this percentage trends downwards as time between releases goes up.

IMPLICATIONS Container adoption among developers continues to increase. The percentage of respondents who say they are currently using containers in their organization has more than doubled since our survey last year. This shift has helped organizations deploy and service applications more quickly by simplifying the decomposition and deployment environments, as we showed in our 2016 Guide to Continuous Delivery.

RECOMMENDATIONS Try modern application container technologies, if you haven't already (and even if you've tried LXC before). Decomposing your app can make it more resilient and robust, allowing for quick recovery from downtime and easier fixes and updates to your application code. To understand how far lightweight resource isolation can take you, read Ivan Dwyer's article "Servers? Where We're Going We Don't Need Servers" on page 26 in this Guide.

THE CLOUD IS EVEN MORE POPULAR FOR PRODUCTION THAN FOR DEVELOPMENT AND TESTING

DATA 62% of survey respondents say they perform production or deployment on a cloud platform; this is compared to 54% who use cloud for development, and only 49% who use cloud for QA or testing. Still, 31% of respondents plan on performing QA/Testing on the cloud, versus 26% who have plans to start deploying to the cloud and 21% who have plans for developing on a cloud platform.

IMPLICATIONS Deploying on cloud platforms can increase availability and make scaling easier. For deployed applications, with real users in hard-to-predict numbers, these benefits become especially important. Developers may currently feel a little more comfortable keeping pre-production applications closer to home, but as more granular as-a-service offerings gain maturity, the cloud will be used increasingly at all application lifecycle stages.

RECOMMENDATIONS Look into cloud platforms to increase application availability and scale elasticity. Consider using cloud services 'farther left.' For a more concrete picture of how to build applications with many-9 SLAs, check out Andreas and Michael Wittig's article *"Building Blocks for Highly Available Systems on AWS"* on page 12 in this Guide.

SECURITY, PERFORMANCE, AND SCALABILITY ARE FAR AND AWAY THE BIGGEST FACTORS THAT AFFECT DEVELOPERS' CHOICE OF CLOUD PROVIDER

DATA 87% of respondents said security was a "very important" factor when choosing a cloud provider, followed by performance (79%), then scalability (73%). The fourth most important factor trailed by fourteen percentage points: 59% said price was "very important."

IMPLICATIONS Cloud pricing can become a significant pain point, but developers are much more concerned about technical and governance factors than price alone. This suggests that service levels are significantly inhomogeneous over all providers (or else price would make more of a difference).

RECOMMENDATIONS Research and plan at a deep technical level before choosing a cloud platform for your application. Be sure to know the platform's strengths and its weaknesses and how well its offerings match your application's needs. For details on how to choose a cloud and move your application to the cloud, see Daniel Bryant's *"Moving to the Cloud: Transforming Technology and the Team"* on page 22 below. For the increased importance of a synoptic view of all application components running in the cloud, read JP Morgenthal's *"Full Stack Engineering in the Age of Hybrid Cloud"* on page 6 below.

KEY RESEARCH FINDINGS

704 IT Professionals responded to DZone's 2016 Cloud Development Survey; the demographics of this survey are as follows:

- 67% of these respondents use Java as their primary programming language at work.
- 76% have been IT professionals for over 10 years.
- 39% work at companies whose headquarters are located in Europe, 35% in the USA.
- 40% work at companies with more than 500 employees, 16% at companies with more than 10,000 employees.

AWS STILL ON TOP

When asked what cloud service providers their organization uses, an overwhelming amount of respondents to DZone's 2016 Cloud survey said they had used Amazon Web Services. 58% of respondents said their org used AWS in some way—more than twice the runner-up, Microsoft Azure (which 26% or respondents said their organization uses). Coming in third were Google Cloud Services, which had 17%.

When cross-tabbed with which issues the respondents expected or experienced with their cloud platform, respondents who used each of these top three service providers were more likely to respond that they "neither experienced nor expected to experience" than the average of all cloud service provider users (608 out of our 704 respondents). See Figure 1 for a breakdown of those providers compared to the average of all other providers.

Respondents expected or experienced greater scalability and higher availability from using cloud platforms over other benefits we asked about (such as better application performance and faster time to market), so it makes sense why orgs might want to stick to larger or more established cloud providers—they're seen as more likely to be able to have more data centers, more availability zones, more robust infrastructures to scale easily and guard against downtime.

Still, there's certainly some stick-to-itiveness involved in which cloud providers companies are using. Between last year's DZone Cloud survey and this year's, AWS usage gained just 1%, while Microsoft gained 2% and Google lost 2%. Providers such as Heroku, Rackspace, and OpenStack barely saw a change (under 1%). Digital Ocean saw the biggest boost, jumping from 5% to 8%.

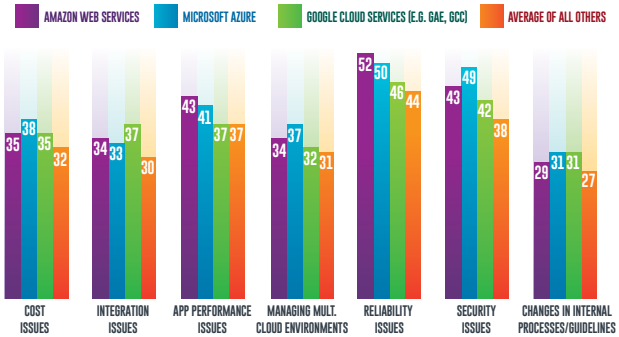
DOCKER IS HUGE. BUT YOU KNEW THAT

When we asked about which open-source cloud products devs used last year, we didn't include Docker. We found OpenStack to be pretty popular (30% in 2015), with Cloud Foundry and OpenShift as runners-up (12% each in 2015). But the majority of our 2015 respondents (55%) said they had not used an open-source cloud product for a business application.

This year, we decided to throw Docker into the mix and add it as an option to the same question. Sure, it's not quite the same as the PaaSes or IaaSes it was pitted against, so we're not going to try to compare the results against other open-source options (apples and oranges, and all that).

45% of our survey respondents this year answered that they had used Docker for a business application. And when asked about whether their org was using or looking into using containers (something we did ask about in 2015), about 66% said their org was either using or evaluating a container technology right now—which is up about 20% from last year's responses, pointing to a lot of recent container adoption that seems to match the buzz surrounding containers and Docker.

01. **CLOUD PLATFORM ISSUES NEITHER EXPERIENCED NOR EXPECTED, BY CLOUD PROVIDER USED**



02. **DOCKER USAGE COMPARED TO TIME BETWEEN RELEASES**

We also found that as developers' time between releases decreased, their container usage increased. While devs with all sorts of release schedules did say they were evaluating containers, those who release "on-demand" were 17% more likely to use containers currently than those who estimated "monthly" releases. Since containers can make decomposing apps easier, they also make it easier to make a change to one part of an app without touching anything in another container, so you have to worry less about patches turning into bugs.

LARGER ORG, MORE HYBRID CLOUD

There are plenty of factors that go into deciding whether to choose a public, private, or hybrid cloud platform: security, availability, scalability, cost, flexibility... the list goes on. For our last three Cloud surveys, we've asked respondents "Which cloud platform type best fits your company's needs." The answers haven't fluctuated much between those surveys. About half of respondents prefer hybrid, while the other half leans slightly toward private.

We found, however, that there is a direct correlation between the size of a respondent's company and which type of cloud platform that respondent thinks best fits that company's needs. Those in small companies (from 1–9 employees) were 15% less likely to choose hybrid cloud platforms than those in the largest companies (10,000+ employees), with a quite linear trend in between. The choice of public cloud platforms trended linearly in the other direction, with 36% of those in the smallest companies choosing public, versus 9% in the largest companies.

Of course, very small companies are generally less likely to be able to handle an all-private cloud infrastructure, just given the number of employees they would need in order to maintain it. Public cloud is—in the absence of dedicated enterprise IT staff—just easier. But that doesn't negate the advantages of having some of your cloud platform private. As cloud computing becomes more widespread, hybrid's popularity will certainly increase; at the same time, as public cloud platforms become easier (and cheaper), it is likely to close the gap with private amongst those who haven't jumped on the hybrid bandwagon.

PAAS LOVES THE LANGUAGES

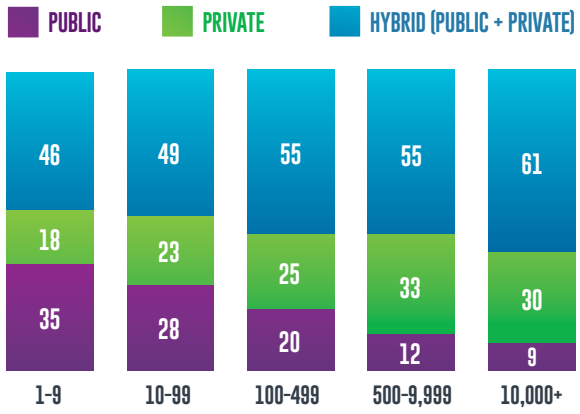
We developed a hypothesis about Platforms as a Service—the most popular cloud-based service in production amongst our respondents. It seemed that, because of how closely PaaSes are tied to the language being deployed, that those respondents whose organizations used fewer languages would be more likely to use a PaaS. Need to use more languages? Maybe you'd go with an IaaS—more work, but more flexibility.

Actually, when correlated against the number of programming languages each respondent said their organization uses(up to four), there was an upward linear trend for PaaS usage. Respondents who said their org only used one language were 49% likely to say their org used a PaaS. Those who said four languages were used at their organization were 68% likely. (Data above four languages for a single org was insufficient for analysis.)

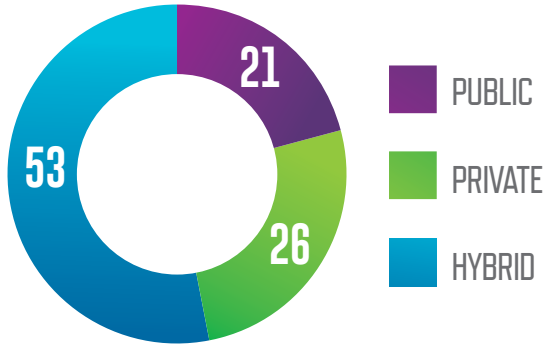
With multiple PaaSes, however, it seems an organization can more easily handle multiple languages. Yes, perhaps a single PaaS is tied to a particular language; but multiple PaaSes would allow an organization to utilize different languages for different uses, without having to build its own platform for each.

For Infrastructures as a Service—the second most popular cloud-based service in production—data was a little more scattered, but still trended upward, up to three languages used. Still, those who only selected one language for their organization were only 49% likely to say their organization uses an IaaS; two-language respondents were 57% likely; three languages showed 64%; and four languages hopped back down to 54%.

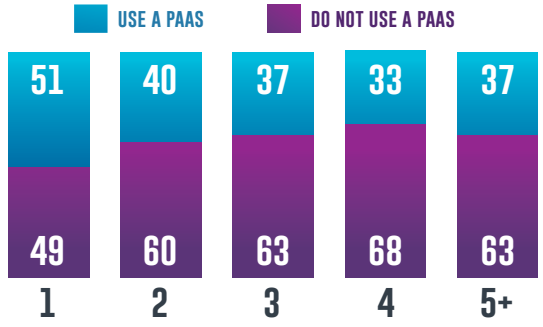
03. PUBLIC, PRIVATE, OR HYBRID CLOUD PREFERENCES, BY COMPANY SIZE



04. PUBLIC, PRIVATE, OR HYBRID CLOUD PREFERENCES OVER ALL RESPONDENTS



05. PAAS USAGE, BY NUMBER OF LANGUAGES USED IN THE ORGANIZATION



Fullstack Engineering in the Age of Hybrid Cloud

QUICK VIEW

01
It's predicted that more than 80% of Enterprise IT organizations will commit to hybrid cloud architectures by 2017.

02
Operating these applications across a variety of cloud service providers will introduce new challenges and complexities.

03
The FSE must now adapt to understand applications operating across resources that they do not directly control.

BY JP MORGENTHAL

The Hybrid cloud describes an architecture that allows businesses to procure platform services from multiple sources. This implies that these services are divided across some physical boundary, each possibly with their own disparate characteristics with regard to use and operation. For the Full Stack Engineer (FSE)—an engineer that has experience understanding the various layers that comprise an application stack inclusive of the physical and logical architectures—hybrid cloud will introduce additional levels of complexity and challenge even the best Full Stack Engineer's capabilities.

In their November 2015 report, "IDC FutureScape: Worldwide Cloud 2016 Predictions — Mastering the Raw Material of Digital Transformation," Analyst firm IDC predicts more than 80% of Enterprise IT organizations will commit to hybrid cloud architectures by 2017. The drivers behind this will be the desire to leverage the best and most compliant platform for a given application component. While a strong business strategy, it will often ignore the consequences of operating these applications across a variety of cloud service providers and, more importantly, performing root cause analysis when something doesn't work as planned.

A recent IDC report entitled "DevOps and the Cost of Downtime: Fortune 1000 Best Practice Metrics Quantified" found that—on average—infrastructure failure costs large enterprises $100,000 per hour. Critical application failures exact a far steeper toll, from $500,000 to $1 million per hour. This is often due to the lack of ownership by any one single group. This difficulty only becomes more problematic when applications are being divided across cloud boundaries.

This leaves many enterprises with the considerable burden of figuring out how to deploy modern or cloud-native applications across multiple architectures and manage the release and operations of these applications. The answer to this problem requires more than just tearing down the silos; it requires employing more full stack engineers.

In today's cloud world, the FSE is someone that understands the nature of a distributed application regardless if components of the application are operating on bare-metal servers in corporate-owned data centers or on public clouds. Hence, they may not be

an expert on every component involved in making that application work, but they understand the basic flow of data across the entire application, the impact of the various components with regard to performance on the application, and which questions to ask of subject matter experts to arrive at an understanding when an application is not operating at its defined service levels.

VALUE OF FSE AND THE CASE OF THE CHATTY APPLICATION

A chatty application is one that tends to flood the network with messages regarding activities. These messages include the application data itself on a data plane, along with management data packets along a control plane. While under most normal conditions the chattiness of the application is not observable by end users, it may result in issues with connectivity and failures due to latency.

Given these outcomes, it can be expected that an end user would probably call the service desk to report their issue. Supposing the resulting ticket gets routed to the network and infrastructure team, they may assess the problem and determine that the chatty application is causing issues and isolate that traffic using quality of service capabilities of the router. If that ticket gets routed to the level 2 engineering support team, they may determine that the application that failed needs to be modified to better deal with latency.

> HYBRID CLOUD WILL INTRODUCE ADDITIONAL LEVELS OF COMPLEXITY AND CHALLENGE EVEN THE BEST FULL STACK ENGINEER CAPABILITIES.

Ultimately, we don't know which solution is the best one. Correcting the issue in the infrastructure means implementing QoS rules that may be operational 100% of the time even though the problem only occurs 20% of the time. Correcting the issue in code means that we now need to maintain specialized code for a unique circumstance that may be highly dependent upon other network events that are occurring, such as a corporate town hall meeting over the intranet.

In the end, the FSE could best assess the conditions under which the problem occurred and would be able to trace the data flow from the network through to the application. The FSE may determine that the application is unnecessarily chatty and the appropriate fix would be to limit messages on the control plane. However, only by understanding the context in which the problem occurred—the end user asserting that system responds with high latency at times—and having the skills to assess the function of the application within that context can one implement the appropriate solution.

> ON AVERAGE, INFRASTRUCTURE FAILURE COSTS LARGE ENTERPRISES $100,000 PER HOUR. CRITICAL APPLICATION FAILURES EXACT A FAR STEEPER TOLL, FROM $500,000 TO $1 MILLION PER HOUR.

WHAT THE FSE NEEDS TO KNOW

Solving the aforementioned chatty application problem requires a certain set of skills that defines the boundaries of a full stack engineer. The following is a sampling of skills that a full stack engineer might have:

- **Networking:** The FSE should have a detailed understanding of IP networking inclusive of routing. Additionally, the FSE should be able to use packet tracing and analysis tools.
- **Servers:** The FSE should have a detailed understanding of server configuration management inclusive of operating systems and hypervisors. This includes both bare-metal and virtual servers, as well as data center tools, such as KVM switching and keyboard logging.
- **Application Infrastructure:** The FSE should have an understanding of how various components of application infrastructure work, such as databases, message queuing, mail servers, web servers, application servers, etc. While it's not feasible for any one individual to know all products in this space, there are certain standards these products implement, such as SMTP, JMS, APMQ, Tomcat, etc., that the FSE should be very knowledgeable about.
- **Data Persistence and Modeling:** An FSE should have understanding for how the data persistence architecture may impact performance. This includes:
  - How the data is physically stored, for example direct-attached storage and Storage Area Network (SAN),

as well as the types of medium used, such as flash, Solid State Drives (SSD), or Hard Disk Drives (HDD).

- How the data is logically represented, such as file or database.
- How the data is organized within the logical format.

- **Data Flow:** For a given application an FSE should be knowledgeable in how the data moves between the various components of the application and across the application infrastructure. This should also account for how the physical architecture is connected.

- **Information Security:** Ideally, the FSE might also be a Certified Information Systems Security Professional (CISSP), which means that they will have a proven understanding of multiple domains surrounding information security. At a minimum, the FSE will need to understand authentication and authorization, firewalling, data loss prevention, and logging.

At this point, the business will have a high likelihood of achieving solid root cause analysis in cases of systems and service outages, and it will review designs for implementation and provide guidance and assurances. Of course, the business could get this same result by hiring multiple individuals with domain expertise in each of these areas; however, even then, they would lose the understanding for how these various domains impact each other. This is the true value of the FSE.

THE FSE IN A HYBRID CLOUD WORLD

As if being an FSE was not complex enough, the introduction of the hybrid cloud architecture has emerged, introducing a whole new level of complexity. To the aforementioned skills, the FSE must now also have understanding for applications operating across resources that they do not directly control. This includes the public internet, shared Infrastructure-as-a-Service (IaaS) and Platform-as-a-Service (PaaS) environments, and cloud services with their assorted Application Programming Interfaces (API) all operated within a by a pay-for-use business model.

This is a real game changer for FSEs in today's world, as they must now attempt to discern new methods of analyzing operational performance, leveraging only what the cloud service providers have made available. In contrast, there is considerably more transparency when the FSE has access to the physical device specifications. Additionally, the FSE is now expected to present estimated costs for operating these applications based on usage projections and service pricing—one of the few skills that is inherently non-technical.

CONCLUSION

In line with current DevOps thinking, the FSE approach raises certain concerns with regard to avoiding the creation of a culture of "heroes." That is, how does an organization develop this capability without having this capability become a single point of responsibility for all systems, and thus become a bottleneck to design and delivery. Scalability will certainly be an issue here as it will be difficult to find many of these resources.

Businesses are going to have to rely on tooling to help with this problem. In this model, the FSEs will need to help develop blueprints for analysts to watch for and identify common issues, pulling in the FSE for decision management. This is akin to surveillance analysts in the military, who are responsible for discerning intelligence from multiple sources and then providing that information to senior military officials who will decide on appropriate action. In this case, the FSE can assess the intelligence gathered by operations analysts.

Sadly, there are no university programs or training courses one can take to become a full stack engineer. Some of the knowledge needed will be learned on the job, but for most who attain this level of understanding, it often comes at the price of personal investments of time and money to acquire and use the technologies an FSE must know. Not surprisingly, most FSEs come from an application development background and learn the other skills as a way to deploy and debug their own applications.

The good news is that learning the skills necessary to be considered a full stack engineer is definitely worth the time and effort. It will be one of the more important requirements as companies drive for applied DevOps principles and attempt to shift left as much of the responsibility for application quality to the development stages. Those with the skills to be considered an FSE will be paid, on average, 20% – 30% more than typical developers and will see easier growth into roles such as Chief Architect and Chief Technology Officer.

JP MORGENTHAL is a Distinguished Engineer with CSC and an internationally renowned thought leader in the areas of digital transformation, IT modernization, cloud computing and software engineering. In 2005, JP founded Avorcor and delivered a multi-tenant cloud platform for API-enabling existing retail, logistics and warehouse management platforms leveraging technologies only now becoming mainstream in 2016, such as microservices, APIs and IoT. JP is sought after for his ability to demonstrate the application of technology for positive business outcomes as well as his ability to represent these ideas and concepts to C-level executives. JP is the author of four trade publications with his most recent being "Cloud Computing: Assessing the Risks."

THE STEP-BY-STEP GUIDE FOR

KEEPING YOUR CLOUD APPLICATIONS SECURE

While cloud adoption is accelerating rapidly, security and data privacy professionals are stretched to the limit keeping up. How do you ensure data in the cloud is secure? Which cloud applications can be trusted with business critical information? Cloud apps reside outside the perimeter, requiring a different breed of security solutions that follow the data, application, and user. **Let's explore a step-by-step cloud protection plan to help protect valuable corporate data from breach and exfiltration.**

STEP 01 DISCOVERY

Are employees using software applications your IT department doesn't know about? Yes they are!

- ☐ Use cloud app detection tools to identify what's being used (by whom and how often) and whether critical corporate data is involved.
- ☐ Consider a Cloud Access Security Broker (CASB); challenge the vendor to provide a Shadow IT Assessment to learn how big a shadow IT problem you are dealing with.

STEP 2 RISK ASSESSMENT

How do you know which apps present the biggest risk? Know which apps to sanction, monitor, or block (the good cloud/bad cloud challenge).

- ☐ Consider a rating system to identify cloud app risk attributes—this will help focus your protection in the right places: bit.ly/cloudcomprisks.
- ☐ Make sure the rating and reporting system can readily upload, anonymize, compress, and cache log data for shadow IT analysis and easily deliver automated risk assessment reports.

STEP 3 USER COACHING

Do all your employees know about common cybercrime tactics? Do your developers and other IT staff know the OWASP Top 10? The Cloud Controls Matrix?

- ☐ Reduce the Gilligan effect (remember Gilligan's Island? Gilligan unknowingly caused damage everywhere he went). There will always be a Gilligan, but security awareness training will lessen the effect.
- ☐ Employ constant reminders and more formalized quarterly training; this is a simple and low cost way to reduce risk of malware.

STEP 4 POLICY ENFORCEMENT

Security policy enforcement must be highly granular and real-time. These can be harder to achieve for cloud applications.

- ☐ Set policy controls based on user activity, using tools and business rules that are content and context aware based on user group, device, location, browser, and agent.
- ☐ Consider using a secure web gateway (on-premises, public cloud, or hybrid), plus an integrated CASB solution with advanced data loss prevention (DLP).

STEP 5 PRIVACY & GOVERNANCE

How are you addressing privacy and governance? Data in the cloud requires unique data-centric security policies.

- ☐ Use appropriate encryption – it's essential in any context. But for cloud security in particular, tokenization (substitution of secure for non-secure data, complementing a secure lookup table) can be especially practical.
- ☐ Make sure encryption doesn't impact application functionality (searching, sorting, etc.). If any of these are made significantly more difficult, users will figure out how to avoid it.

STEP 6 ENCRYPTED TRAFFIC MANAGEMENT

How are you maintaining privacy while selectively decrypting for security reasons?

- ☐ Consider addressing SSL and other forms of encrypted traffic to/from the cloud.
- ☐ Measure the impact of SSL/TLS on application performance and on payload visibility to internal security professionals.
- ☐ Tip: For industries whose traffic is more than 50% encrypted (like financial services and healthcare), policy-based traffic decryption may require a dedicated SSL visibility subsystem and/or specialized network architecture.

STEP 7 INCIDENT RESPONSE

How do you identify and quickly respond to malicious activity?

- ☐ Avoid the "car alarm syndrome!" Too many false alarms and alerts can be "needles hiding in needles" in the haystack.
- ☐ Consider beefing up incident response with a dedicated forensics function. Malicious software deliberately hides its tracks, and low-level complexity introduced by cloud deployment makes intuitive human interfaces especially key for incident response (e.g. security analytics, free-form search, and integration with 3rd party SIEM systems).

Even if you're not ready to make a big investment in a CASB solution, it's time to start managing cloud risk. To learn more about how to move to the cloud with confidence, view this webcast: dc.bluecoat.com/Forrester_Webinar

BY LISA ROM, SENIOR MANAGER OF AMERICAS MARKETING, BLUE COAT SYSTEMS

MICROSERVICES

PATTERNS FOR BUILDING MODERN APPLICATIONS

Iron.io's microservices architecture, delivered via Docker containers, enables the world's most innovative companies to transform large data sets into job processing that drives their revenue.

> *Over half (55%) of all organizations are planning to deploy this type of architecture in the future, and 28% in the next twelve months.*
> *- IDG Connect*

www.iron.io

What Will it Take to Have a Serverless Future?

The advancements in cloud technologies continue to provide developers with the capabilities to build and ship applications quicker and more effectively. The next evolution of the cloud lies in even more intelligent systems that can react and adapt to internal and external events with automated workflows. To fully realize this serverless future, there will be a convergence between a few ongoing trends:

- **MICROSERVICES**
  Decoupling application components as individual services that perform a single responsibility enables workload independence and portability
- **CONTAINERS**
  Building, deploying and running code within a lightweight, containerized runtime ensures consistency from development to production
- **EVENT-DRIVEN COMPUTING**
  Breaking apart from the traditional request/response model allows workflows to react to dynamic environments automatically
- **DEVOPS**
  Automated resource provisioning, configuration, and management brings it all together at the infrastructure layer to abstract away everything but the API

Contrary to how it may sound, though, the holy grail of DevOps isn't NoOps. The real point is to make operations such a natural extension to the development process that it doesn't require a special skill set to run code at scale in any environment. The continued evolution of cloud infrastructure services and development platforms are enabling this type of serverless future, where developers can build and deploy distributed applications with confidence.

WRITTEN BY DAVE NUGENT
DEVELOPER EVANGELIST LEAD, IRON.IO

PARTNER SPOTLIGHT

Container-based Job Processing BY IRON.IO

Iron.io

"Hybrid cloud [is] used by 38% of organizations today ... by the end of 2016, two thirds (66%) of organizations expect to be utilizing [hybrid]." -IDG

CATEGORY
Enterprise job processing

NEW RELEASES
Continuous

OPEN SOURCE?
Yes

CASE STUDY

Second only to ESPN, Bleacher Report is a growing force for team-specific sports content. Bleacher Report delivers real-time sports scores, news, and team updates via the Team Stream mobile app. With over 12 million downloads of the Team Stream app, users get a personalized fan experience wherever they are.

The Bleacher Report ops team has long held 99.9% uptime as a primary goal. In order to achieve this goal, changes to the infrastructure had to be made.

"Because our system was already worker based, it was easy to experiment with Iron.io and we had a functioning prototype copy of our existing system in less than a day," recalls Eddie Dombrowski, Sr Software Engineer. "A week later and we were able to migrate our code base to use Iron.io."

STRENGTHS

- Any cloud deployment, public, private, or hybrid
- Scalable processing workers as a service
- Serverless developer experience
- Ability to incorporate existing security policies
- Increased speed and agility through containerized microservices

NOTABLE CUSTOMERS

- Twitter
- Google
- Turner
- Zenefits
- Whole Foods

BLOG iron.io/blog | TWITTER @getiron | WEBSITE iron.io

Building Blocks for Highly Available Systems on AWS

QUICK VIEW

01 AWS offers services like RDS, DynamoDB, and S3 which let you build highly available systems.

02 Decoupling your services by using load balancers or queues is another building block for HA architecture.

03 AWS offers data centers in different regions worldwide with Availability Zones, which distribute your workload and allow you to recover even if a whole data center fails.

BY ANDREAS WITTIG & MICHAEL WITTIG

High availability describes a system operating with almost no downtime. Even if a failure occurs, the system is able to recover automatically—although a short service interruption might be necessary. When talking about high availability, we refer to the AEC-2 classification of Harvard Research Group: "minimally interrupted computing services . . . during most hours of the day and most days of the week throughout the year."

UPTIME MATTERS

Downtimes are painful. Customers aren't able to place orders in your online shop, access your content, or interact with your customer support team. An outage causes direct (e.g. less orders) and indirect (e.g. loss of trust) costs. It is also a stressful situation for everyone involved in operating a system.

Building highly available systems was expensive in the past. For example, distributing a system across isolated data centers resulted in noticeable costs, as a second data center involved purchasing floor space, racks, hardware, and software.

Thanks to the services and infrastructure AWS is offering, operating highly available systems has become much more affordable. That's why highly available systems are becoming the new standard even for start-ups, small, and mid-sized companies.

A highly available system reduces risks for your business and avoids burned out operations engineers.

HIGH AVAILABILITY ON AWS

AWS offers more than 70 different services. Some of them offer high availability by default:

- SQL database (RDS with Multi-AZ deployment)
- No-SQL database (DynamoDB)
- Object storage (S3)
- Message queue (SQS)
- Load balancer (ELB)
- DNS (Route 53)
- Content Delivery Network (CloudFront)

The availability of your system depends on its weakest part. Whenever you are adding a new AWS service to your architecture, ask yourself the following questions:

1. Is this service highly available by default?
2. If not, how can I use the service to be able to recover from a failure automatically?

There should always be a positive answer at acceptable costs to one of these two questions.

EC2 IS NOT HIGHLY AVAILABLE BY DEFAULT

High availability is a key principle of AWS. Nevertheless, Werner Vogels, CTO of Amazon.com, is quoted with saying "Everything fails, all the time." This doesn't imply AWS is offering unreliable services. The opposite is true. The quote visualizes how AWS is treating failure: by planning for it.

> HIGHLY AVAILABLE SYSTEMS ARE BECOMING THE NEW STANDARD EVEN FOR START-UPS, SMALL, AND MID-SIZED COMPANIES.

An important part of AWS is not highly available by default: virtual machines (EC2 instances). A virtual machine might fail because of issues with the host system or due to networking issues. By default, a failed EC2 instance is not replaced, but AWS offers tools to recover from failures automatically.

AWS offers data centers in different regions worldwide. Each region consists of at least two isolated data centers, called Availability Zones. Distributing your workload on multiple servers in at least two Availability Zones allows you to recover even if a whole data center fails.

Launching EC2 instances in multiple Availability Zones is easy and comes at no extra cost. You are able to do so manually or by using a tool called Auto Scaling Groups. An Auto Scaling Group automates the creation of multiple EC2 instances based on the same blueprint. It also allows you to evenly distribute a fleet of servers across multiple Availability Zones.

1ST PREREQUISITE FOR HIGH AVAILABILITY: STATELESS SERVER

To be able to spread your workload on multiple servers, you need to get rid of state stored on a single server. If your EC2 instances are stateless, you can replace them whenever you need to without losing data.

Your applications probably need to store data—but where do you store your data if not on your EC2 instances? Outsourcing your data to storage services might help. As mentioned, AWS offers the following data stores:

- SQL database (RDS with Multi-AZ deployment)
- No-SQL database (DynamoDB)
- Object storage (S3)

All these storage services offer high availability by default. So you are able to use them as a building block without introducing a single-point-of-failure.

If your legacy applications are storing data on disk, using one of those three storage options is impossible. Currently there are two options:

- Use the beta service Elastic File System, which offers storage available through NFS.
- Synchronize state between your EC2 instances (e.g. with distributed file systems like GlusterFS).

Unfortunately there is no production-ready, out-of-the-box service available from AWS right now.

2ND PREREQUISITE FOR HIGH AVAILABILITY: LOOSE COUPLING

Another prerequisite is to decouple your virtual machines from incoming requests. To be able to distribute your workload on multiple, automatically replaced EC2 instances, you are in need of a reliable and static entry point into your system.

Depending on whether incoming requests need to be processed synchronously or asynchronously there are two different services available on AWS to act as an entry point into your system: a load balancer (ELB) for synchronous requests or a queue (SQS) for asynchronous requests. Both are common building blocks for a highly available system on AWS.

> A HIGHLY AVAILABLE SYSTEM REDUCES RISKS FOR YOUR BUSINESS AND AVOIDS BURNED OUT OPERATIONS ENGINEERS.

CONCLUSION

AWS offers infrastructure and services that you can use as building blocks for highly available systems. Compared to on-premise environments, the risk of an outage can be reduced dramatically without the need of a big budget. That's why high availability is becoming the new standard for systems running in the cloud.

ANDREAS WITTIG is a Cloud Specialist focusing on AWS and DevOps. He is author of Amazon Web Services in Action (Manning). He helps his clients gain value from AWS. As a software engineer he develops cloud-native applications. He migrated the complete IT infrastructure of the first bank in Germany to AWS. Andreas is constantly creating new products and services based on bleeding edge technology to keep on learning.

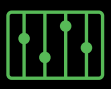
MICHAEL WITTIG is author of Amazon Web Services in Action (Manning). He helps his clients gain value from AWS. As a software engineer he develops cloud-native real-time web and mobile applications. He migrated the complete IT infrastructure of the first bank in Germany to AWS. He has expertise in distributed system development and architecture, with experience in algorithmic trading and real-time analytics.

NGINX

Migrating to the cloud?

Let us be your guide.

Advanced load balancing and caching



Custom health checks for high availability

Visibility into the performance of your infrastructure

Built and optimized to run on cloud and containers

Use independently or with load balancing services

Learn more at:
nginx.com/cloud

5 Tips for Optimal Cloud Performance

The cloud helps you get applications up quickly - but you still need to optimize your infrastructure if you want high-performance applications. NGINX has built the leading cloud-native application delivery software, so we've assembled these five tips to help you get the speed you need.

TIP 1. SIZE RESOURCES TO FIT

Sharing physical resources, such as RAM, with other people's VMs can cause unpredictable performance slowdowns. To reduce contention, use a dedicated host, or size your VM CPU, storage, and other requirements to use entire actual physical resources. Then use the following tips to make your application as hardware-independent and resilient as possible.

TIP 2. USE A REVERSE PROXY SERVER

Use a reverse proxy server to interface with the Internet and a separate application server to deliver your site's services, then optimize each function. This speeds up overall performance and provides benefits such as security controls and monitoring capabilities.

TIP 3. USE MULTIPLE APP SERVERS WITH LOAD BALANCING

Use your reverse proxy server as a load balancer for multiple application servers. The number of servers scales dynamically to match user demand, cutting costs. You can use NGINX load balancing alongside cloud capabilities such as Elastic Load Balancing on Amazon Web Services.

TIP 4. CACHE STATIC AND DYNAMIC CONTENT

Cache static files – GIFs, JPEGs, CSS files, and so on – on the reverse proxy server. Then cache application-generated files briefly, for instance for one second, to give your application server a big break.

TIP 5. MEASURE AND MONITOR

Measure performance and monitor uptime and responsiveness. Then, experiment with each aspect of your implementation to optimize it.

Read more of our blog posts to learn about cloud-native flawless cloud-native application delivery at scale.

WRITTEN BY FLOYD EARL SMITH
TECHNICAL MARKETING WRITER, NGINX

PARTNER SPOTLIGHT

NGINX Plus BY NGINX

NGINX

NGINX Plus is the complete application delivery platform for the modern web.

CATEGORY
Application Delivery

NEW RELEASES
2-3 major releases per year with regular updates between

OPEN SOURCE?
Open Source & Commercial

CASE STUDY

MuleSoft offers a leading cloud-based integration platform that enables enterprises to easily and securely connect apps, data, and devices. MuleSoft's challenge? Meet customer performance and availability requirements, with little visibility into the customers' applications or upstream configurations. MuleSoft relies on NGINX Plus for load balancing, monitoring, and more. NGINX Plus enables MuleSoft to accurately forward requests to thousands of upstream workers with minimal latency. In addition, with NGINX Plus MuleSoft can easily ensure end-to-end SSL encryption of customer traffic. NGINX Plus gives MuleSoft flexibility in its cloud platform to innovate faster and provides them with the scale and performance they need.

STRENGTHS

- HTTP, TCP, and UDP load balancing
- Easy to deploy in any environment
- Advanced monitoring and management tools
- Rich security controls

NOTABLE CUSTOMERS

- Airbnb
- AppNexus
- Discovery Education
- Groupon
- Netflix
- Uber
- Wix
- Zendesk
- Blue Jeans

BLOG nginx.com/blog | **TWITTER** @nginx | **WEBSITE** nginx.com/cloud

Architecting for Failure

BY GAURAV PURANDARE

QUICK VIEW

01
There are all kinds of failures you need to account for in your application design when working in cloud environments.

02
Isolating individual application components reduces overall complexity and provides clarity on resources utilized.

03
Make it a point to learn the lessons each failure offers to prevent the issue from recurring.

The design of all credible, fault-tolerant architectures is based on one extremely important principle: "Anything that can go wrong, will go wrong." - Murphy's Law

Murphy's Law makes it necessary to consider failures an integral part of any system, something that can be particularly tricky in cloud environments because users expect near-100 percent uptime. Today, there are many organizations that embrace the cloud in the belief that they are automatically shielded from infrastructure-related issues, or that those issues are "just handled" by their cloud provider without impacting an application's functionality. Although this may be the case sometimes, there are many failures you still need to account for in your application design.

ISOLATE COMPONENTS

As the level of complexity in applications continues to increase through new layers of abstraction and the introduction of cutting edge technology, isolation becomes a powerful aid in anticipating and mitigating failures. Isolating individual application components to different systems not only reduces overall complexity, but it also provides clarity on the resources utilized. Maxed-out resources are often an indicator for (imminent) failure, and isolation can help pinpoint such issues. Further, when component isolation is combined with replication, it prevents applications from having single points of failure, adding resilience and making failures less likely overall.

DETERMINE THE REASONS FOR FAILURES

HARDWARE AND INFRASTRUCTURE

Even though the risk of hardware failure might appear to be a dwindling issue, it cannot be ignored altogether. Applications should always be developed to be prepared for hardware failures.

For example, having sticky sessions between a load balancer (LB) and the servers sitting behind that LB causes all traffic for a specific user session to be routed to a specific server. In the event of this server failing, a user would find the application or service unavailable.

Paradoxically, rapid success can breed failure as well. An application that experiences rapid adoption may not be able to cope with the corresponding server or networking load. If an application supports auto-scaling, then all implicated servers should be configured to start all necessary services at system boot up in order to scale out when demand spikes. Regular health checks (industry best practice is to do this every five minutes or less) within and via the LB infrastructure become essential.

Similarly, if failure occurs with a database server (assume the worst case scenario, such as the primary or master server being impacted), there needs to be a configuration in place to elect a new primary or master node. Application libraries need to adjust their database connections, which ideally should be an automated process.

SOFTWARE AND CONFIGURATION

Apart from expected, well-known and well-understood issues (such as resource over-utilization), the application should be able to cope with systemic and global changes. A classic example here is the leap second bug from 2012, which caused

many issues for Linux servers. Today, applications are prepared for this issue, so the latest leap second (in 2015) had a much smoother transition. Although no one can predict when the next undiscovered issue of this magnitude will strike, by following major industry blogs and key influencers on Twitter, you can discover and address new issues swiftly once they have been identified.

Sometimes it is neither the hardware, nor the application, but ancillary configuration that can cause failure. Perhaps it's a simple firewall rule change, or the lack of a static IP/address for a specific component that is referenced by several others. This is exacerbated when multiple people or teams collaborate on an application. A useful strategy to avoid such issues is to restrict access to critical rules and configuration to specified individuals that are both trusted and have a holistic understanding of the overall environment.

BE PROACTIVE

Having proper monitoring in place will help you proactively determine possible failures before they become an issue. Build scripts to perform daily sanity tests and automate port scanning and access controls, so you're not wasting time and won't get sloppy with essential (but repetitive) tasks. That way, you only need to pay attention if your script issues an alert.

However, be sure to also monitor the monitoring scripts themselves. If your monitoring system doesn't show a status of "all green," we know we have a problem. However, if it shows all green across the board, it is still a best practice to confirm that there isn't a more nefarious issue with the monitoring system itself and to verify that the correct metrics and parameters are being monitored.

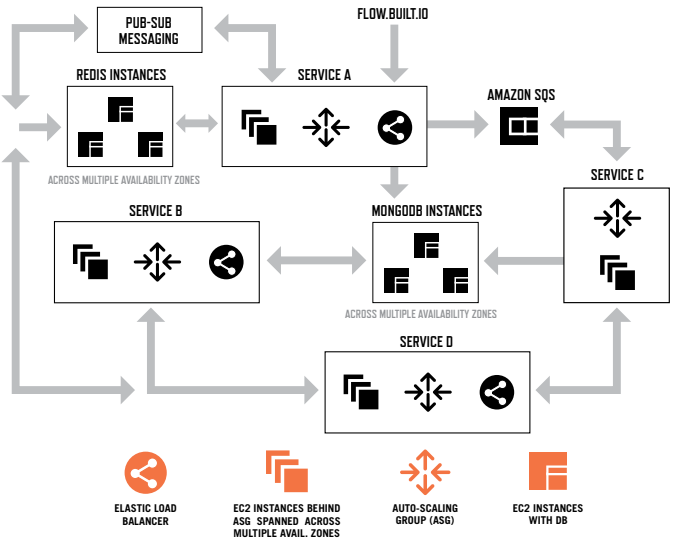
REHEARSE FAILURE

Replicate your critical systems and then simulate failure. Review error messages and check the remedial actions documented on your recovery checklists. Try to determine the maximum scalability of different components in your system. While such measures might seem like overkill for a casual application, they are a worthwhile investment for all mission critical ones. This particular precaution won't necessarily decrease your failure rate, but it will help you detect and respond to issues faster, whenever they occur on a key production system.

NEVER STOP LEARNING, NEVER STOP TUNING

Even with the best architecture, the most reliable technologies, and optimal operations, failures—big and small—will occur. When they do, make it a point to learn the lessons each failure offers and to prevent the issue from recurring. Perform a root cause analysis. Discuss failures with engineering, with QA, with DevOps, and with the team consuming the app or service that was ultimately impacted. Update your checklists and use each failure to make yourself smarter and your systems more resilient.

CLOUD ARCHITECTURE BLUEPRINT



To give an example of an architecture designed for failure, the above diagram illustrates a high-level architecture based on Built.io Flow. All internet-facing services make use of an Elastic Load Balancer (ELB) layer to balance traffic across a pool of Amazon EC2 instances associated with a corresponding auto-scaling group. These EC2 instances span across multiple Availability Zones, thus avoiding downtimes for zone failures.

Because of built-in redundancy, a connection failure to a database instance like Redis or MongoDB is handled gracefully by the application. If connectivity issues are observed for a primary/master database server, for example, the application can immediately reconnect to a newly elected primary/master server.

Remember: design for failure so things don't fail.

BEST PRACTICES	TOOLS/TECHNIQUES
ISOLATE COMPONENTS	• Review architecture diagrams • Replicate components
HARDWARE AND INFRASTRUCTURE	Perform stress testing using tools like Loader.io
SOFTWARE AND CONFIGURATION	Validate application configurations (e.g., if a config file is stored in JSON, validate it using JSONLint or an equivalent tool)
BE PROACTIVE	Build port scanning scripts, Cloud-Checkr works great for AWS
REHEARSE FAILURE	Test your system by stopping or deleting a random component. For larger operations, consider deploying a tool like Chaos Monkey

GAURAV PURANDARE is Senior DevOps Engineer at Built.io, a technology provider with solutions enabling organizations to quickly create, integrate, and scale apps across web, mobile, and IoT. His passion for architecting cloud applications pairs with his years of experience managing and operating multi-tier, highly available applications. He is a certified AWS Solution Architect – Associate level and holds certified MPN Competency for Microsoft Azure.

DIVING DEEPER

INTO CLOUD DEVELOPMENT

TOP 10 #CLOUD TWITTER FEEDS

 @BGRACELY

 @KRISHNAN

 @RANDYBIAS

 @JEFFBARR

 @JAMESURQUHART

 @THECLOUDNETWORK

 @ARCHIMEDIUS

 @WERNER

 @UTOLLWI

 @KEVIN_JACKSON

DZONE CLOUD-RELATED ZONES

Cloud Zone

dzone.com/cloud

The Cloud Zone covers the host of providers and utilities that make cloud computing possible and push the limits (and savings) with which we can deploy, store, and host applications in a flexible, elastic manner. This Zone focuses on PaaS, infrastructures, security, scalability, and hosting servers.

Integration Zone

dzone.com/integration

Enterprise Integration is a huge problem space for developers, and with so many different technologies to choose from, finding the most elegant solution can be tricky. The EI Zone focuses on communication architectures, message brokers, enterprise applications, ESBs, integration protocols, web services, service-oriented architecture (SOA), message-oriented middleware (MOM), and API management.

IoT Zone

dzone.com/iot

The Internet of Things (IoT) Zone features all aspects of this multifaceted technology movement. Here you'll find information related to IoT, including Machine to Machine (M2M), real-time data, fog computing, haptics, open distributed computing, and other hot topics. The IoT Zone goes beyond home automation to include wearables, business- oriented technology, and more.

TOP CLOUD REFCARDZ

Getting Started with OpenStack
bit.ly/startopenstack

Cloud Foundry
bit.ly/startcloudfoundry

Essential PostgreSQL
bit.ly/startcloudcomputing

TOP CLOUD WEBSITES

Cloud Harmony
cloudharmony.com

Netflix Tech Blog
techblog.netflix.com

Cloud Tech News
cloudcomputing-news.net

TOP CLOUD RESOURCES

AWS Whitepapers

Understanding Cloud Computing Vulnerabilities

International Journal of Cloud Computing

PAAS TO MODEL-DRIVEN PAAS:

How Enterprises Can Accelerate Time to Business Value

According to Gartner's Hype Cycle, we have passed the top of the hype around Platform-as-a-Service, which means it is a good moment to assess PaaS adoption in the enterprise.

But first, we need to clarify what we mean with PaaS. In general, we can distinguish 3 layers of PaaS:

- **Foundational PaaS:** the layer in which we stop talking about infrastructure and switch to an app-centric view of the world.
- **PaaS:** platforms that take code as input and run it as an app (aPaaS), platforms that take integration flows as input and execute them (iPaaS), or platforms that store and query your data (dbPaaS).
- **Model-Driven PaaS:** platforms that take models as input and execute them. They are also called High-Productivity aPaaS (Gartner) or low-code platforms (Forrester).

If you look at these different PaaS types, you would think that Model-Driven PaaS follows PaaS adoption and is more of a niche market. In the enterprise, the opposite is true. The simple reason is that Model-Driven PaaS delivers business value quicker.

PaaS adoption is driven by the need for digital innovation. In order to innovate, companies must become software companies, with faster, more agile software delivery cycles. PaaS makes deployment and operations more efficient but lacks a vision for making app development faster.

That's where Model-Driven PaaS comes in. Model-driven development enables companies to get to market 6 times faster and with 70% less resources than code-based approaches. It also provides a common language to engage the business, improving user acceptance and project success rates.

Enterprises see the need for digital innovation. They are changing the way they deliver software, focusing on creating smaller, cross-functional teams and shorter feedback cycles. For enterprises, their platform of choice is low-code, model-driven and highly productive.

WRITTEN BY JOHAN DEN HAAN
CTO AT MENDIX

PARTNER SPOTLIGHT

Mendix Platform

mx mendix

One unified platform that brings IT and business together to transform ideas into applications

CATEGORY
aPaaS, High Productivity

NEW RELEASES
Monthly

OPEN SOURCE?
Yes

CASE STUDY

LV= Insurance adopted Mendix to pursue initiatives that weren't making it to the top of the agenda because the time and cost of doing it the traditional way wasn't worth it. The company is now rapidly delivering new applications that generate major cost savings and help sustain its growth in a competitive market.

Projects include the LV= Broker Portal, which was delivered in 15 days, and end-to-end insurance products with capabilities spanning customer self-service, administration, claims and fulfillment. By taking a modular approach and building reusable components, LV= can now launch new products in just 6 weeks.

This approach has saved LV= an estimated £5 million since 2012 while earning it the coveted Celent Model Insurer Award.

STRENGTHS

- Platform facilitates IT-business collaboration, improving project success rates.
- Visual model-driven development delivers 6x productivity over hand-coding.
- Full app delivery cycle support: design, build, test, deploy, manage and iterate.
- Open platform enables easy integration with any system, cloud service or data source.
- 1-click deployment on major cloud platforms (AWS, Azure, Bluemix, HP Helion, Pivotal).

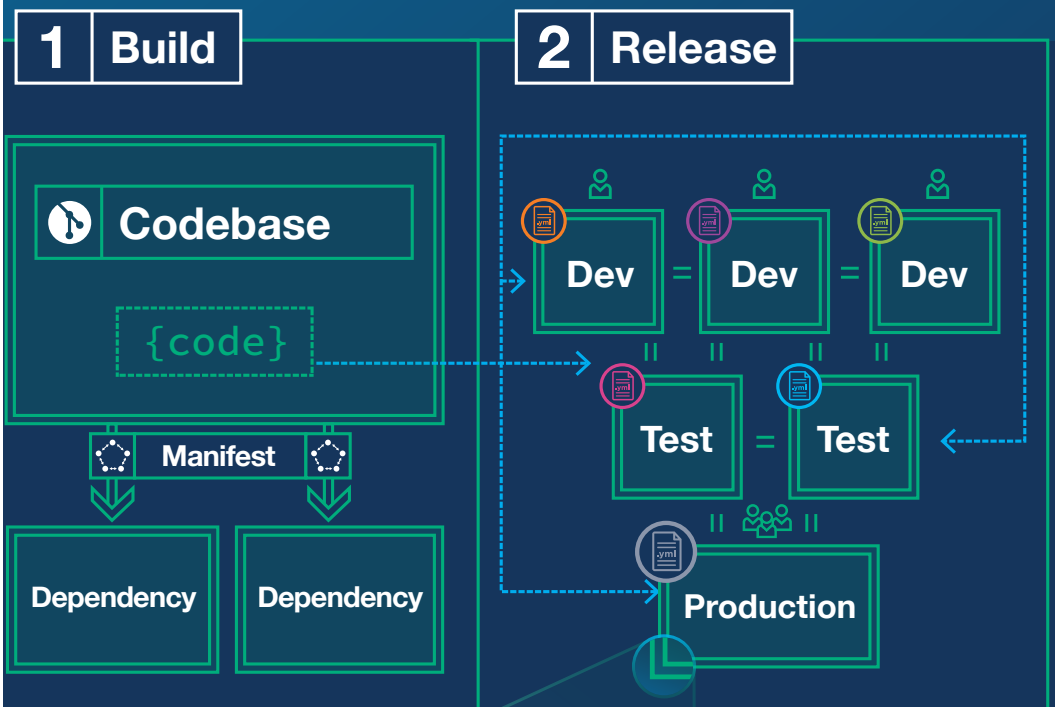
NOTABLE CUSTOMERS

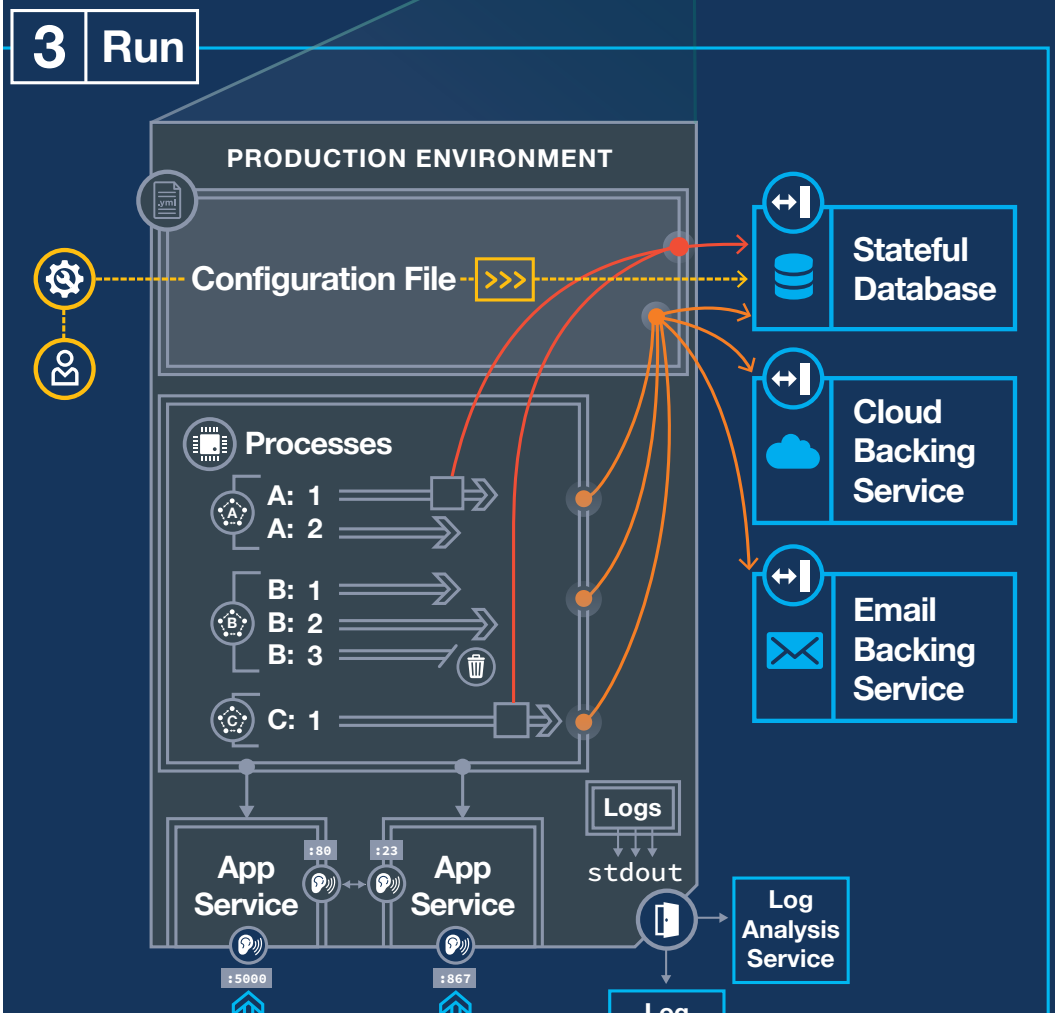
- ABN Amro
- ING
- Sprint
- Chubb Group
- Merck
- TNT
- Erie Insurance
- MIT

BLOG mendix.com/blogs | **TWITTER** @mendix | **WEBSITE** mendix.com

The 12-Factor App

Modern web applications run in heterogeneous environments, scale elastically, update frequently, and depend on independently deployed backing services. Modern application architectures and development practices must be designed accordingly. The PaaS-masters at Heroku summarized lessons learned from building hundreds of cloud-native applications into the twelve factors visualized below.





the 12 factors

Codebase
One codebase, many deploys, strict version control

Dependencies
Explicitly declare and isolate dependencies

Configuration
Store config in each deploy environment, preferably using environmental variables

Backing Services
Treat backing services as resources (neutral as to local vs. third-party) located via config

Build, Release, Run
Strictly separate build, release, and run; never change code at runtime

Processes
Keep all processes stateless and share-nothing; store state (with other persistent data) in a stateful backing service

Port Binding
Bind every service to a port and listen on that port; don't rely on runtime server injection

Concurrency
Distinguish process types (e.g. web, background worker, utility) and scale each type independently

Disposability
Make processes start up quickly (<4s from launch to ready) and shut down safely (for web process: stop listening, finish current requests, exit; for worker process: return current job to work queue)

Dev/Prod Parity
Maximize dev/prod parity by minimizing gaps in time (between deploys: hours), personnel (authors=deployers), and environment (use adapters for backing services)

Logs
Log by writing all output streams to stdout; rout streams using non-app services

Admin Processes

MOVING TO THE CLOUD: Transforming Technology and the Team

BY DANIEL BRYANT

QUICK VIEW

01
Share clear goals, define measures of success (KPIs), set communication expectations.

02
Start small with your cloud migration, chose a vendor based on your use cases, be aware of cloud geography, and design 'cloud-native' applications.

03
Don't forget the people and the organizational transformation required to support a cloud migration.

Developing new software for—or migrating existing applications to, the cloud is inherently a transformative process. As we have written about previously, the approach to architecture and design must change, and topics, such as distributed computing and fault tolerance, become critical. As with any transformative process within a business, there are some fundamental processes that must be established.

The first is setting clear goals. If you are working as a developer or architect, it is essential that the goal of moving to the cloud is clear for you. Is it cost reduction, a decrease in time-to-market for products, or an increased ability to innovate? The second process required is defining measures of success—i.e., how do we know we are making progress? Are we saving money week over week? Are we deploying more applications per month, or have we run more experiments this quarter? The final process is establishing clear expectations of communication. For example, do we report current work within a daily standup, what progress reports should be delivered, and how do we communicate issues?

A (CLOUD-BASED) JOURNEY OF A THOUSAND MILES, BEGINS WITH...

START SMALL

When an organization is attempting to introduce cloud computing, small experiments should be conducted first. In our consulting work at OpenCredo we often recommend creating a simple proof of concept (POC) based on a subset of business functionality,

GOAL	METHODS
COST SAVINGS	• Scaling compute resource to real-time demand (for both user-generated traffic and batch tasks), which removes need to over-provision • Reduced TCO of infrastructure (no hidden costs of maintenance staff; no data center insurance, electricity, or cooling)
INCREASED INNOVATION	• Compute resources for experiments can be acquired on demand • Experiments can be implemented at scale (the cost of which would be prohibitive to run on-premise) • Access to cutting edge 'as-a-service' technology, e.g. machine learning platforms
DECREASED TIME-TO-MARKET	• Compute resources can be acquired on-demand without the need for capacity planning • Environments can be replicated on demand for testing or staging, or for entering new geographic markets

which should be conducted separately from 'business as usual' (BAU) activities. We have also recommended the use of 'hack days' (or a 'hack week', depending on scope) where an IT or software development team is given free reign to experiment with the new technology available.

Criteria for POC

- Business functionality is limited to a single department. This limits communication and governance overhead, e.g. a new payment service.
- Required functionality is provided as a 'vertical' (self-contained) application with minimal external integration points, e.g. a new customer sign-up site/page.
- Buy-in is achieved with the team responsible for the work.

The key goal of this stage within the plan is for the IT team to become familiar with the cloud paradigm (for example, we still frequently see the look of amazement when developers realize they can spin up and access powerful hardware within seconds), which paves the way for the rest of the business to adapt to the cloud paradigm (e.g. different billing mechanisms and approval processes), and enables the evaluation of the various cloud platforms in relation to the typical use cases within the business.

ALL CLOUDS ARE NOT CREATED EQUAL—CHOOSE WISELY

Each cloud platform has inherent properties, strengths, and weaknesses. For example, Amazon Web Services (AWS) offers a breadth of services, but the orchestration of components required to deliver the specified functionality can at times be complex. Google Cloud Platform (GCP) offers a narrower, more clearly-defined range of functionality, but the offerings are more typically opinionated. The strong ('big') data processing focus of GCP was a major contributor for Spotify migrating their infrastructure to this platform. Microsoft Azure has a strong set of offerings and is often a compelling platform choice if the existing organization is heavily dependent on the Microsoft stack.

The primary goal for this stage of a cloud migration is to determine and catalogue the primary use cases that will be implemented within the initial six to twelve months of work, and then map these requirements to the offerings from the various cloud vendors.

BE AWARE OF GEOGRAPHY

The geographical range of the cloud is often a key enabler. No longer does expansion into a new geographical market take years of planning and infrastructure contract negotiations. Now, we simply select a different 'region' from our cloud vendor's user interface, deploy our required stack, and we are good to go. The flipside of this ease of deployment is that care must be taken in regards to where data flows and is stored at rest. Data sovereignty is often a critical component of many business operations, and accidentally sending data across international borders for information processing may violate regulations and compliance mandates.

Any plan to migrate applications to the cloud must include an approach to data requirements ('due diligence'), and the legal and regulatory constraints must clearly be communicated to the IT team and across the business. In addition, deployment of applications across multiple geographic regions and availability zones is highly recommended in order to mitigate the risk of unavailability and data loss.

TUNING APPS TO THE CLOUD

Much is currently being written about the benefits of creating software systems as clusters of 'microservices.' Regardless of how small your services are, any software that is deployed to the cloud will benefit from adhering to many of the same technical goals, best codified as 'Twelve-Factor' applications. Obviously, not all existing applications can be migrated to this format—the requirement for 'stateless' applications, in particular, is troublesome for many applications written before the advent of cloud technologies. However, the fact that almost all components within a cloud will be communicating over a network means that developers should become familiar with fundamental and advanced distributed computing principles.

The completion of this stage within a cloud migration plan should result in a catalogue of applications to be migrated (or created), with the associated technical, 'non-functional' requirements or constraints clearly enumerated. Validation of all functional and nonfunctional requirements must be included within an automated build pipeline. In relation to the previous section of this article, it is worth paying special attention to applications communicating across 'availability zones' (essentially datacenters) and across geographical regions. What may appear as a small gap in an architecture diagram can add significant latency or potential for failure.

DON'T FORGET THE ORGANIZATION (AND THE PEOPLE!)

ORGANIZATIONAL DESIGN AND TRANSFORMATION

Often the first sign of organizational design struggles appears as a team moves from the proof-of-concept to the implementation phase of cloud migration. As soon as implementation expands beyond one team, then the complexity of interactions increases. This can often be challenging on a technical level, but it is almost always challenging on an organizational level. Look for red flags like queues of work, long delays, or 'sign offs' on work as it moves around an organization, as well as for teams pulling in different directions (or using competing technologies).

We often work with the senior management C-level within organizations, as—unless alignment and buy-in is achieved at this level—any changes made within the rest of organization can easily unravel.

THE IMPORTANCE OF DEVOPS

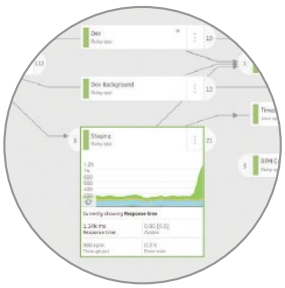
Although the term 'DevOps' has become somewhat overused (and still isn't truly defined), we believe that the concepts behind it—such as (1) a shared understanding and responsibility across development and operations; (2) automation, with principles and practices driving tooling, not the other way around; and (3) creating signals for rapid feedback—are vital for success within a cloud architecture. As the number of application components is typically higher within a cloud-based application (in comparison with more traditional platforms), we often see problems emerge rapidly when teams are misaligned on goals, solving the same problem in multiple different ways; when cargo-culting of automation, or the incorrect use of off-the-shelf 'DevOps tooling' occurs; and when an absence of situational awareness is rampant.

We have seen DevOps implementations create fear within organizations, and we have also seen suboptimal processes being automated (automated failure is still failure!). We believe that the concepts and goals behind the 'DevOps' movement are vital in the wider business context and the current economic climate, where time to market and speed of innovation are clear competitive advantages.

DANIEL BRYANT (@danielbryantuk) is Chief Scientist at OpenCredo, a software consultancy and delivery company dedicated to helping clients deliver better software faster. Currently, Daniel specializes in continuous delivery, DevOps methodologies, and leveraging container-based platforms such as Docker, Mesos and Kubernetes. He is also a leader within the London Java Community, and a contributor to the Java Community Process (JCP).

One source of truth.

See all your data. Boost performance. Drive accountability for everyone.



IT Operations
Faster delivery.
Fewer bottlenecks.
More stability.

Mobile Developers
End-to-end visibility,
24/7 alerting, and
crash analysis.

Front-end Developers
Reduce resolution times
and spend more time
writing new code.

App Owners
Track engagement.
Pinpoint issues.
Optimize usability.

Simple. Powerful. Secure.
Find out why New Relic helps you build and run great software at
newrelic.com/why

New Relic.

3 Keys to Cloud Monitoring Success

The cloud provides many benefits to dev and ops teams, including no infrastructure to build or maintain, faster rollout of new app versions, and greater elasticity.

These practices—critical to companies that aspire to compete with software—require your monitoring tools to adapt to a new reality of very transient workloads running on highly modular architectures. These three capabilities need to be top-of-mind when evaluating a monitoring system for the cloud.

1. LOW FRICTION

One thing many users love about the cloud is just how easy it is to get started. Your monitoring solution should be no different. Ideally, you should be able to begin monitoring virtual server loud computing cinstances such as Amazon's EC2, often in as little as one minute. In addition, when you build new applications, they should be instrumented with a monitoring system with instant-on availability so that there is no lag between deployment and visibility.

2. FULL-STACK VISIBILITY

When a cloud application issue strikes, the clock is ticking. To quickly narrow down the scope of a problem, you need to be able to rapidly discern trends and changes across both the cloud services and your code. Your monitoring solution should be able to marry data from your cloud provider, the underlying operating system, and the application or service running on top to provide a complete view of health and status.

3. DYNAMIC WORKLOAD VISIBILITY

Since virtual server instances like Amazon EC2 can last a very short time, it's easy for the list of instances in your monitoring solution to quickly fill up with a lot of "gray servers" that no longer report health and status. With physical servers, this is useful information—perhaps someone unplugged the server's power or network connection, and you might want to stroll over to the data center and check things out. But for cloud servers, seeing a gray server often doesn't present any useful information. Bottom line: it's important for your monitoring solution to automatically remove this cruft.

WRITTEN BY ABNER GERMANOW
SR. DIRECTOR, STRATEGIC CAMPAIGNS & EVANGELISM, NEW RELIC

PARTNER SPOTLIGHT

New Relic Software Analytics Cloud BY NEW RELIC

New Relic.

Shift, re-fit or modernize. No matter your approach, New Relic gives you the data you need to make your efforts a success, with 100% visibility into your entire stack.

CATEGORY
APM

NEW RELEASES
Daily

OPEN SOURCE?
No

CASE STUDY

Founded in 2006, Veracode's automated, cloud-based service safeguards web, mobile, and cloud applications for more than 800 organizations worldwide, including three of the top four banks in the Fortune 100. The company deployed New Relic to monitor both its customer-facing platform and internal business systems, but an area where the software analytics solution particularly shines is in helping Veracode make the best use of public cloud resources. "We use New Relic and the data we get from Amazon Web Services to get a full picture of application health and performance in the cloud," says Patrick Hetherton, director of SaaS operations at Veracode. "Plus the insight we get from New Relic helps us be cost-effective with our cloud usage."

STRENGTHS

- Performance monitoring across applications, browsers, cloud infrastructure, and more
- Customer experience management for web and mobile channels
- Proactive root cause analysis anywhere in the stack
- Extensible platform offering partner integrations, open APIs, and 100+ plugins
- Secure, multi-tenant SaaS architecture delivering value out of the box within minutes

NOTABLE CUSTOMERS

- Hearst
- Trulia
- Lending Club
- MercadoLibre
- MLBAM
- Tapjoy
- HauteLook and Nordstromrack.com

BLOG blog.newrelic.com | **TWITTER** @newrelic | **WEBSITE** www.newrelic.com

Servers?

Where We're Going We Don't Need Servers.

BY IVAN DWYER

QUICK VIEW

01
The convergence of microservices, containers, event-driven patterns, and DevOps automation lay the foundation for a serverless architecture.

02
The next evolution of developer empowerment, this further abstracts away the underlying operations to power workloads at any scale.

03
The best practices here will let you build highly reactive systems, responding to the ever-changing environments of the modern world with highly streamlined, automated workflows.

"I DON'T KNOW HOW TO TELL YOU THIS, BUT YOU'RE IN A TIME MACHINE."

Advancements in application architecture patterns all have a core purpose: developer empowerment. In today's fast-paced and ultra-competitive world (where every company has to be a software company), organizations are doing all they can to enable their developers to ship applications faster. With such high expectations, what could possibly be more promising than a serverless world?

Despite the implications, serverless computing doesn't mean getting rid of the data center through some form of black magic that powers compute cycles in thin air. At its core, the concept promises a serverless experience for developers, who never have to think about provisioning or managing infrastructure resources to power workloads at any scale. This is done by decoupling backend jobs as independent microservices that run through an automated workflow when a predetermined event occurs. These events arise from a variety of sources—from an application, as with a webhook; in the real world, as with a sensor capture; within a system, as with a database change; or on a schedule, as with a cron job. The event then triggers the workflow behind the scenes—spin up, execute, tear down. Rinse and repeat at massive scale. This form of event-driven computing shifts configuration from the systems layer to the application layer, thus turning the DevOps mantra *Infrastructure as Code* into *Infrastructure in Response to Code*.

"THERE'S SOMETHING VERY FAMILIAR ABOUT ALL THIS."

Cloud computing has come a long way in recent years. Virtualized infrastructure resources with elastic auto-scaling capabilities, platforms that abstract operations for delivery and management, along with a wide range of complementary toolkits, have all come together to provide developers with a highly effective environment for building large scale distributed applications.

With these continued advancements across the entire cloud stack, what makes this serverless trend any different?

The primary difference lies in the nature of the workloads. Simply put, an application behaves different than a job. When we push an application to the cloud, we do so thinking about where it will live and how it will run. This is because it has a known IP address and an open port to accept incoming requests. On the other hand, when we build a job, we do so only thinking about when it will execute. In breaking away from the traditional request/response model towards an event-driven model, automated workflows react to dynamic environments accordingly. Even though there is compute involved, it's completely outside of the development lifecycle, thus making the paradigm "serverless."

In order for such laissez-faire development to pass even the most basic smoke test, there must first be a discrete unit of compute that we're confident is consistent from development to production. Container technologies such as Docker provide a lightweight runtime environment that isolates job processes with its dependencies, clearly specified through a standard packaging format. Compared to VMs, which have a wider scope, containers provide only what is needed for each individual job, minimizing its footprint and ensuring its consistency. If we then follow the commonly accepted characteristics of microservices when writing our code—loosely coupled, stateless services that each perform a single responsibility—what we're left with is a collection independent and portable workloads that can be executed at will without the need for oversight. Fire away.

"WHAT ABOUT ALL THAT TALK ABOUT SCREWING UP FUTURE EVENTS?"

While event-driven computing patterns have existed for some time, the serverless trend really caught on with the introduction

of AWS Lambda in late 2014. Microsoft has a similar offer with WebJobs, and Google recently announced its version with Google Functions. For a solution independent of any sole infrastructure provider, Iron.io offers a container-based serverless computing platform that is available in any cloud, public or private.

Lest we forget: the more abstraction put forth, the more activity happening behind the scenes. To actually reap the benefits of a serverless architecture, one must fully grasp the software development lifecycle and underlying operations to avoid it becoming a hapless architecture. The following is an introduction to the process of building Docker-based serverless jobs along with some best practices to help get you started.

BUILDING THE JOB

Developing with Docker is a breeze, as you can work across multiple languages and environments without clutter or conflict. When your job is ready to build, you specify the runtime by writing a Dockerfile that sets the executable, dependencies, and any additional configuration needed for the process.

- Choose a lightweight base layer. This can be a minimal Linux distribution such as Alpine or Busybox.
- Keep the layers to a minimum. Don't run an update on the OS and consolidate `RUN` operations inline using `&&` when possible.
- Limit the external dependencies to only what's needed for the process itself, and vendor ahead of time so there's no additional importing when the job is started.

UPLOADING THE JOB IMAGE

Each serverless job is built as a Docker image and uploaded to a registry, where it can be pulled on demand. This can be a third party public image repository such as Docker Hub, Quay.io, or your own private registry.

- Incorporate the job code into a CI/CD pipeline, building the container image and uploading to a repository.
- Version your images using Docker Tags and document properly. Don't rely on `:latest` as what should always run.

SETTING EVENT TRIGGERS

With such a potentially wide range of event sources, there can be a tendency for the associated jobs to pile up quickly. It is crucial to set the triggers properly to ensure the right workflows are kicked off and that no data is lost in the process.

- Map each job to your API (at a minimum within your documentation, but you can also set endpoints for direct requests). Using an API Gateway is a common way to manage events and endpoints across systems.
- Use a load balancer for synchronous requests and a message queue for asynchronous requests to throttle and buffer requests when load is high.

CONFIGURING THE RUNTIME ENVIRONMENT

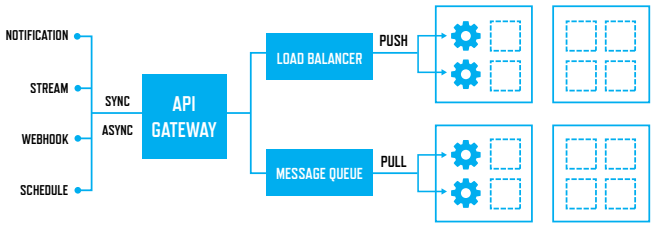
Operational complexities such as service registration and container orchestration are abstracted away from the development lifecycle; however it is not recommended to just "set it and forget it" when running in a production environment.

- Profile your workloads for their most optimal compute environment. For example, some workloads are more memory intensive and need more memory allocated.
- Set how many concurrent jobs can execute at any given time. This can help keep costs down and ensure you don't overload the system.
- Determine what happens when the job fails. If you want to auto-retry, set the maximum number of times with a delay in between.

SECURING AND MONITORING THE JOB

To be production-grade, wrapping the environment with proper security and monitoring is essential. Given the levels of abstraction that serverless computing provides, it's even more important to gain insight into what's happening behind the scenes.

- Payload data should be encrypted at the source and then decrypted within the job code itself. Public key is a common technique in this scenario.
- Connections to databases from within a job process that are outside the network should be secure, either through a VPN or IP whitelisting.
- Inspect `stdout` and `stderr` for each job process. You can pipe these logs to `syslog` or a 3rd party logging service.
- Maintain a real-time dashboard of all queued, running, failed, and finished jobs.



End-to-end lifecycle of a serverless workload

"WHAT HAPPENS IN THE FUTURE?"

With this serverless computing trend, the gap between the infrastructure layer and the application layer narrows even further. A well orchestrated container fleet combined with a well choreographed set of workloads leads to more intelligent systems across the board. The event-driven patterns set forth in this article provide developers and architects a way to respond to the ever-changing environments of the modern world, where everything from our bodies to the planet is connected. The next evolution in cloud computing will stem from these patterns to create predictive systems that can learn and adapt accordingly. A workload-aware global compute cloud that knows what, when, and where best to run workloads is the ultimate vision for developer empowerment in this modern cloud era. It won't take a flux capacitor to get there—we're already on our way as an ecosystem to enable this future.

IVAN DWYER is the head of business development at Iron.io, collaborating with ecosystem partners across the entire cloud stack to form meaningful strategic and technical alliances. Iron.io is a leading provider of serverless computing for the modern Enterprise, with its Docker-based job processing platform.

CloudBees

Jenkins World 2016

The event for everything Jenkins: community, CloudBees and ecosystem.

Jenkins World will bring together the largest gathering of Jenkins users in the world, including Jenkins experts, continuous delivery thought leaders and companies offering complementary technologies for Jenkins. Jenkins World will provide opportunities for attendees to learn, explore and network face-to-face, as well as to help shape the future of Jenkins.

Santa Clara Convention Center
September 13–15
Register now: www.jenkinsworld.com

CloudBees
SOASTA Performance is Everything
CPRIME
JFrog
SAUCELABS
{ } Codenvy
MOBILE LABS

Enabling Continuous Delivery as a Service

Continuous delivery (CD) is the practice of automatically and continuously building, testing and deploying software. CD is an extension of continuous integration (CI) and provides the foundation for DevOps. CD enables IT organizations to reduce time to market, increase software quality, lower costs and gain competitive advantage.

Small IT organizations, the decision to run CD on a public cloud is easy; they leverage the cloud heavily to reduce costs and increase agility. They are able to run CD as a Service – to instantly provision resources for teams and scale-up/down as needed. In fact, Jenkins, a popular open source automation server, is available in pre-built machine images for Amazon EC2 and Microsoft Azure.

For enterprises, concerns around security and compliance represent a barrier to using the public cloud for CD. Executing CD on a public cloud entails provisioning access to valuable code and assets. Therefore, many enterprises host public-facing infrastructure on public clouds and host back-end sensitive code, services and databases on internal private clouds. However, CD infrastructure is also a great workload for private clouds. CD as a Service implemented on a private cloud lowers costs, provides the benefits of elasticity and scalability and also secures valuable assets.

The **CloudBees Jenkins Platform - Private SaaS Edition** is an elastic, scalable Jenkins-based platform that enables organizations to offer CD as a Service on an OpenStack private cloud or on an AWS Virtual Private Cloud with an interface to the datacenter.

CloudBees Jenkins Platform - Private SaaS Edition benefits three key roles:

- **Shared services manager:** Set-up a fault tolerant Jenkins as a Service on your private cloud in minutes
- **Jenkins administrator:** Self-service access to the CloudBees Jenkins Platform for teams, enabling instant onboarding of new Jenkins projects
- **Developers:** Jenkins as a Service accelerates software delivery by providing immediate access to resources

Bringing together proven cloud technologies with the power of the CloudBees Jenkins Platform, Private SaaS Edition accelerates adoption of CD and DevOps, while addressing security concerns and providing the advantages of the cloud. Learn more.

WRITTEN BY KALYAN [KAL] VISSA
SR. PRODUCT MANAGER, CLOUDBEES

PARTNER SPOTLIGHT

CloudBees Jenkins Platform

BY CLOUDBEES, INC.

CloudBees
The Enterprise Jenkins Company

Based on the time-tested, proven, extensible, and popular Jenkins automation platform. Advanced CD Pipeline execution and management.

CATEGORY
DevOps Automation Platform

NEW RELEASES
Bi-annually

OPEN SOURCE?
Yes

CASE STUDY

Morningstar Inc. - Leading provider of independent investment research

CHALLENGE: Improve build processes while laying the groundwork for an organizational move to continuous delivery and a DevOps culture

SOLUTION: Use continuous integration with the CloudBees Jenkins Platform to increase automation, improve consistency, lower administrative overhead, and facilitate the adoption of continuous delivery practices and a DevOps culture

RESULTS:
- Administration time and overhead reduced 80%
- Time to resolve typical support issues reduced by 70%
- Release cycles shortened

STRENGTHS
- Enable Continous Delivery as a Service for your teams.
- Self-service access to Jenkins for all project teams.
- Deploy Jenkins servers in seconds.
- Fast installation on private cloud.
- Elasticity and High Availability for Jenkins.

NOTABLE CUSTOMERS
- Netflix
- Nokia
- TD Band
- Orbitz
- Lufthansa
- Apple

BLOG blog.cloudbees.com | TWITTER @cloudbees | WEBSITE www.cloudbees.com

Focusing on the Cloud in Cloud Applications

QUICK VIEW

01
The move to cloud applications is making developers take a harder look at their networks and how they affect performance.

02
IaaS, microservices, and composable applications are powering the transition to new network requirements.

03
Cloud applications must make different assumptions about key variables such as loss, latency, and reliability.

BY NICK KEPHART

Remember TCP slow start? CIDR notation? For a long time networking knowledge has been pushed to the back of most developers' minds. But recently, a networking renaissance has kicked off with developers leading the charge. Major services are flirting with HTTP/2 for pipelining and multiplexing. Some mobile developers are even building their own transport stacks to control application experience delivered to end users on high-latency or low-bandwidth connections. Developers are jumping back into the networking fray.

Let's talk about the Ops in DevOps for a moment. Brush up on networking, because as you re-architect your applications for IaaS and microservices, the network is becoming an increasingly critical enabler, or inhibitor, of your application experience. Your users are ever more dispersed across networks, just as application services are spread over more and more IaaS regions, API endpoints, and external services. You don't have to look far to find name brand cloud applications that have been knocked out by network failures, from Netflix to Craigslist, Salesforce to Playstation.

THE CHANGING STRUCTURE OF APPS AND THE NETWORK

Three trends that are reshaping application development are also changing the relationship between application and network.

IaaS: As cloud-based infrastructure becomes more common, application flows increasingly traverse public Internet links. Hosting your site on AWS? Azure? Softlayer? Google? Each provider has different peering policies, global POPs, and traffic management that can lead to dramatically different end-user experiences. Google peers widely, meaning that it is rarely more than one or two networks away from your users. AWS and Microsoft, on the other hand, focus on backhauling traffic across or between continents on their own backbone to have greater control.

Microservices: Developers are increasingly slicing up application services into ephemeral, scale-out workloads. Often based on Linux containers using iptables-based NAT, these services are causing an explosion in intra- and inter-data center network traffic. Within the data center, this causes more variable network I/O and a drive to smarter routing policies, reshaping the switching, routing, and load-balancing landscape. Outside the data center, this is creating more strain on peering links and requires careful planning for key inter-DC routes.

Composable apps: Most importantly, applications are being composed of multiple parts. We've already seen a rise in APIs, from payments to chat-based support. Some of the APIs are full-fledged services. Take Lumberyard, the new game engine from AWS; developers will find themselves increasingly composing applications from diverse publishers' disparate

network locations. I've seen this happen even within the enterprise data center, with integrations surrounding the Salesforce and ServiceNow ecosystems that have become critical business services.

DELIVERING YOUR BITS

How do these changes manifest themselves? They directly impact variables that your users and your operations teams care about.

- **Increasing loss:** With traffic traversing best-effort Internet links, pay special attention to network health. Be prepared to adapt routing or peering preferences based on observed loss in the network, and regularly monitor the performance of your ISPs or CDNs.
- **Increasing latency:** As traffic travels long distances between API endpoints, IaaS data centers, and individual containers, your app needs to handle highly variable conditions. Test out message queues, API calls, and critical transactions across a range of tolerances.
- **Less reliability:** Develop your application to gracefully handle outages of critical services caused by device failures, route leaks, or DNS problems. Recent network and infrastructure outages have taken down entire services for hours, from popular IaaS providers to large CDNs.

DEVELOPERS MEETING THE NETWORKING CHALLENGE

Cloud-native architectures get a lot of attention, but you can go deeper on the network side too. Interestingly, many developers are beginning to take on this challenge to optimize the delivery of their applications. Here are a few examples of how you might consider improving your own application experience. The right course of action, if any, will of course depend on your applications and users.

Get network savvy: You can get a surprisingly rich view of how your application is delivered over the network. This isn't just your grandfather's ping. New monitoring techniques, such as detailed network path tracing and active probing, identify specific network segments and service providers that have performance degradation. Sophisticated services like ThousandEyes can provide a detailed view of both web and network performance across data centers, ISPs, CDNs, and IaaS providers. There should be no excuse for being in the dark about network performance.

Get automated: Having visibility into both application and network performance is step one. You'll want to make sure that you tie your monitoring into your on-call ticket apps, like PagerDuty or xMatters, with carefully crafted escalation rules to ensure you aren't unnecessarily awakened from your beauty sleep. I've seen some teams go even further and use alerts to power load balancers or change DNS records that can take racks, pods, or entire data centers out of rotation if performance is degraded.

Try new content delivery strategies: Content delivery networks can host a wide variety of content and make it quickly accessible to a global audience. Of course, this means you have another service provider to manage. Some applications, such as Netflix, have built out their own CDN to minimize the network distance to its users. For applications that rely on audio or video streaming—which is particularly sensitive to packet delay and loss, misordered arrival, and jittery TCP round-trip times—a CDN, whether built or bought, can make a huge difference.

> YOU DON'T HAVE TO LOOK FAR TO FIND NAME BRAND CLOUD APPLICATIONS THAT HAVE BEEN KNOCKED OUT BY NETWORK FAILURES

Experiment with network architectures: Some developers aren't satisfied with the network services that they can procure from major providers. So they've built their own. Riot Games unveiled their architecture of new POPs, dark fiber leases, tweaked routing preferences, and peering with major ISPs across the U.S. The result? Higher percentages of League of Legends players experienced under 80ms latency, a desirable threshold for multiplayer gaming. Tying in SDN to control this architecture, and Riot Games expects to push the boundaries even further.

If you dare, build new network stacks: With users on high-latency and low-bandwidth networks, sometimes the standard HTTP/TCP/IP stack doesn't cut it. Facebook's mobile app team shipped their latest Android app for developing countries with its own custom messaging protocol over TLS and TCP, rather than using HTTPS. This reduces data use on 2G networks and works with image servers in the Facebook CDN to deliver exact-sized image delivery that otherwise would eat up tons of bandwidth. This is a great tutorial on building the network stack.

MAKING YOUR CLOUD APP-READY, AND VICE VERSA

Application development today has a rich set of services and infrastructure that can be used to construct ever-more powerful applications. But this has left many applications more exposed to the whims of the network. Using new network monitoring techniques and architectures, developers can better control the end-user experience. So break out an O'Reilly book, chat with your favorite Ops member, and refresh your networking knowledge. It's time to get your cloud app-ready, and your app truly cloud-ready.

NICK KEPHART leads Product Marketing at ThousandEyes, which develops Network Intelligence software, where he reports on Internet health and digs into the causes of outages that impact important online services. Prior to ThousandEyes, Nick worked to promote new approaches to cloud application architectures and automation while at cloud management firm RightScale.

5 Things Developers Need from the Cloud

Developers have high expectations from cloud services. Since I'm a developer, I'm probably biased, but as we are typically the ones spending the most time working with cloud providers and services, our needs are pretty important. Here are five of those needs, ranked in completely random order.

1) PROPER DOCUMENTATION

While the benefits of good documentation may obvious, not all providers deliver – and you can certainly tell when documentation was considered a lower priority for a cloud provider. Your docs should be clear, direct, with little to no marketing fluff. Users should not require a login just to read your docs. I want to see how your service works before I sign up.

2) A COMMAND LINE TOOL

Provide a command line tool that can be installed via npm and easily updated. If you don't use npm, make it easy to find the download link and provide clear instructions on how to update the CLI. To make this even easier, the tool itself should prompt developers when new versions exist.

3) A DEVELOPER PORTAL

This goes hand in hand with the CLI. Anything I do with the CLI should be available via a web-based portal as well. Not all developers prefer the CLI, so providing multiple ways of accomplishing tasks gives the developer options. The portal should also make it easy for me to track usage and billing information.

4) DON'T HIDE/OBFUSCATE PRICING INFORMATION

While developers care more about the technical aspects of your service, we also want it to be easy to figure out what we can expect to pay. If your payment policies are difficult to explain, then that's a good sign that you should look into simplifying your offerings.

5) PLAYS WELL WITH OTHERS

Your service may be cool. It may be the best thing since sliced bread. But your service isn't the only one out there. Make it easy to integrate with other services, even your competition. Make it easy to leave your service if need be. That may seem crazy, but as a developer, I want to know I can change my mind about your service with little to no pain in migration.

WRITTEN BY RAYMOND CAMDEN
STRONGLOOP / NODE.JS DEVELOPER EVANGELIST, IBM

PARTNER SPOTLIGHT

API Connect BY STRONGLOOP AND IBM

IBM API Connect is a complete solution that addresses all aspects of the API lifecycle - Create Run, Manage, Secure - for both on-premises and cloud environments.

CATEGORY	NEW RELEASES	OPEN SOURCE?
API Management	TBA	No

API LIFECYCLE

IBM API Connect offers features to manage the API lifecycle, including:

Create—create high-quality, scalable and secure APIs for application servers, databases, enterprise service buses (ESB) and mainframes in minutes.

Run—take advantage of integrated tooling to build, debug and deploy APIs and microservices using the Node.js or Java.

Manage—create and manage portals that allow developers to quickly discover and consume APIs and securely access enterprise data, and monitor APIs to improve performance.

Secure—Administrators can manage security and governance over APIs and the microservices. IT can set and enforce API policies to secure back-end information assets and comply with governance and regulatory mandates.

STRENGTHS

- Simplify discovery of enterprise systems of record for automated API creation
- Provide self-service access for internal and third-party developers through a market-leading gateway
- Ensure security and governance across the API lifecycle
- Unify management of Node.js and Java microservice applications
- Increase flexibility with hybrid cloud deployment

FEATURES

- Unified Console
- Quickly run APIs and microservices
- Manage API's with ease
- Readily secure APIs and microservices
- Create API's in minutes

BLOG **developer.ibm.com/answers/topics/apim/** | TWITTER **@ibmapiconnect** | WEBSITE **ibm.com/software/products/en/api-connectapim/**

Executive Insights on Developing and Deploying Applications on the Cloud

QUICK VIEW

01
Build platform-agnostic cloud-based apps to ensure they will work anywhere and prevent being locked-in to a particular cloud.

02
Scalability, security, and the ability to control costs are the key reasons why companies are going to cloud-based apps.

03
Cloud-based applications will be at the forefront of connecting all data—video, voice, IoT—every device will be connected.

BY TOM SMITH

In order to gather insights on developing and deploying apps on the cloud we interviewed 28 executives, from 23 companies, actively involved in cloud-based applications for their own companies or their clients. All of the executives have extensive experience developing and deploying applications on the cloud. Specifically we spoke to:

Neeraj Gupta, *S.V.P. Product &Engineering*, Apcera | **Jad Naous**, *Product Lead*, AppDynamics | **Ez Natarajan**, *V.P. Head Cloud Services Business Unit*, Beyondsoft | **Alon Girmonsky**, *CEO and Founder*, BlazeMeter | **Kunal Bharati**, *Cloud Architect* and **Nishant Patel**, *CTO*, Built.io | **Sacha Labourey**, *CEO*, Cloudbees | **Deirdre Mahon**, *CMO* and **Fraser McKay**, *V.P. of Products*, Cloud Cruiser | **Flint Brenton**, *CEO*, CollabNet | **Ali Din**, *Senior V.P.* and *CMO* and **Walid Elemary**, *V.P. Product Development*, dinCloud | **Mike Masterson**, *Director of Strategic Business Development*, Dynatrace | **Gabe Monroy**, *CTO* and **Jasen Hansen**, *Chief Architect*, Engine Yard | **Fred Simon**, *Co-Founder and Chief Architect*, JFrog | **Jim Frey**, *V.P. of Products* and **Ian Pye**, *Co-Founder and Principal Engineer*, Kentik | **Johan den Haan**, *CTO*, Mendix | **Mounil Patel**, *V.P. Strategic Field Engagement*, Mimecast | **Arvind Mehrotra**, *President and Global Business Head – Infrastructure Management Services*, NIIT Technologies | **Faisal Memon**, *Product Manager*, NGINX | **Jens Eckels**, *Director, PaaS Business Group*, Oracle | **Pat Harper**, *SVP Operations* and **Sara Pilling**, *V.P. Strategic Communications*, PGi | **Joan Wrabetz**, *CTO*, Quali | **Partha Seetala**, *CTO*, Robin Systems | **Nick Kephart**, *Senior Director Product Marketing*, ThousandEyes | **Kiran Bondalapati**, *CTO and Co-Founder*, ZeroStack

Here's what we learned from the executives:

01

The companies we spoke with tend to be **agnostic with regards to frameworks, platforms, functionality, software, tools, languages, and cloud-types**. They know they have to be able to work with what their clients are using. What they use varies by the layer of the stack they're working with. The most frequently used languages are Java, Ruby on Rails, and Go—with Go becoming the prevalent programming language for cloud software.

02

The most important elements of cloud-based applications are **scalability, security, cost savings, integration/connectivity, and stability/availability**. The ability to scale up or down depending on consumer needs and wants is critical from a reliability standpoint, as well as from a cost perspective. Anything added to the cloud stack is able to scale both vertically and horizontally. Security has gained importance as companies capture more consumer data with personally sensitive information. The cloud is a means to an end, it provides quick and endless innovation for companies.

03

Multiple problems are being solved by cloud-based applications. **The cloud enables small companies to compete efficiently by establishing processes that allow them to move at warp speed with extreme reliability**. Complex applications are able to integrate with each other. Orchestration makes everything work together on a common platform, enabling companies to solve problems and make smarter business decisions. Customer-facing cloud-based apps can be changed very quickly as business requirements change or as monitoring dictates.

04

Among those we interviewed, there was little agreement with regards to the skills that make someone good at developing and deploying applications on the cloud. **Knowledge of business requirements, realizing business value, ability to solve problems, a desire to learn, and knowledge of scalability and security** were the skills that were mentioned most frequently. However, more than 20 other skills were mentioned, and we're sure the list will continue to evolve.

05

The biggest evolution in the development and deployment of cloud-based applications has been the **move to continuous delivery, automation, containers, and microservices**. Now all processes and handoffs are automated. We are close to completing the process of committing code to run in production in an automated way. Developers are creating cloud-based apps built on microservices with more complex functionality in conjunction with business people developing business processes on the platform. There is much wider distribution with more frequent deployments in shorter time frames due to the virtualization of environments and increased automation.

06

The biggest obstacles to success developing and deploying applications on the cloud are: **1) the complexity of the cloud; 2) philosophical differences; 3) ability to scale; and 4) DevOps and automation**. Most companies don't understand the complexity of the cloud, the multi-tenancy, scalability, and security aspects. Problems are more exacerbated for apps in the cloud. Legacy apps that were barely maintained do not meet the native app requirements of the cloud. There's a philosophical resistance to change. The network is moving to where the company is no longer in direct control. Scale is critical, as are performance, availability, speed and redundancy. You must be able to scale in an automated way. The DevOps footprint has made a staggering impact on the cost and time to delivery; however, few enterprises have adopted a true DevOps mentality or methodology.

07

The biggest concerns regarding the development and deployment of cloud-based applications are **security**; the **speed** with which new technology is emerging and change is taking place; and the **inability to just move legacy apps to the cloud**. New technology is emerging rapidly, allowing companies to adopt—and customers to enjoy—quickly. There are a lot of moving parts and complexity to keep up with. There are no standards with regards to orchestration and security. In two years we'll end up with a clear winner to manage containers and deployments. It's a little risky to decide on a framework today. Security on the cloud is superior to security at the enterprise level, yet there are still regulatory and compliance issues with which you must contend. Trying to move legacy apps to the cloud is not always the best fit—you need to change to cloud-based architecture and deployment.

08

The future for cloud-based applications is very bright, with a lot of visionary ideas. The three themes we heard revolve around **the continued ubiquity and scale of the cloud**, driven by **microservices** and **the increased emphasis on security** as consumers become more savvy about the precarious nature of their data in cloud-based apps. On-premises and cloud-based apps will continue to co-exist in a cloud- and device-agnostic world, and the majority of apps will be in the cloud. Cloud-based apps will continue to disrupt business processes. Microservices and products like Amazon Lambda will enable the evolution of the platform as a service, allowing it to handle the running, implementation, and management of apps, empowering developers to focus on writing code. Customers will feel more confident on public clouds with acceptable levels of security and customization, as there will be more security, trust, and legal requirements regarding personal data that is collected.

09

Three things for developers to keep in mind when working with applications for the cloud: **security, scalability, and performance**. Keep security on the top of your mind as more and more apps get hacked. Security, trust, and reliability go hand in hand. It's also important to isolate each tenant's data and protect sensitive data. Design for scale and performance. The app must be able to handle large spikes during certain times of the day or month. Push to the cloud for fast iteration, frequent deployments, and quality apps.

10

Parting thoughts. When we asked respondents what we failed to ask regarding cloud-based applications, here's what they told us:

- **Not every cloud vendor is the same**. Companies should look at vendors who provide easy, continuous integration/delivery environments, rapid spin-up and tear-down for testing, and transparent updates of infrastructure.
- Containerize—**containers facilitate portability** and allow you to work in every type of cloud.
- **Share what you know**. Everyone benefits from open collaboration.
- **Know where the budget will come from**; costs can rise as the application scales quickly.
- **Re-architect from a microservices standpoint** to help move apps, manage them, and give you more connectivity in more places.
- **Developers will still need to program and collaborate**. The entire team needs to move to a collaboration orientation.
- **Test for failure**—this is one of the models of Netflix.
- **Commit to lifelong learning** and you'll always be able to use what you learn.

TOM SMITH is a Research Analyst at DZone who excels at gathering insights from analytics—both quantitative and qualitative—to drive business results. His passion is sharing information of value to help people succeed. In his spare time, you can find him either eating at Chipotle or working out at the gym.

SOLUTIONS DIRECTORY

This directory contains databases and database performance tools to help you store, organize, and query the data you need. It provides free trial data and product category information gathered from vendor websites and project pages. Solutions are selected for inclusion based on several impartial criteria, including solution maturity, technical innovativeness, relevance, and data availability.

PRODUCT	CLASSIFICATION	PUBLIC / PRIVATE	FREE TRIAL	WEBSITE
ActiveState Stackato	Virtual Machines, App Containers, aPaaS, DBaaS	Private by provider, Private on-premise, Hybrid	Limited by storage	activestate.com
Amazon EC2	Virtual Machines	Public	750 hours, Limited by storage	aws.amazon.com
Anypoint Platform by MuleSoft	iPaaS	Public, Hybrid, On-premise	Available by request	mulesoft.com
Anypresence	aPaaS, mBaaS	Public, Private by provider, Private on-premise	Pilot program available	anypresence.com
Apache CloudStack	Virtual Machines	Public, Private by provider	Open Source	cloudstack.apache.org
Apprenda	App Containers, aPaaS, DBaaS, MBaaS	Private on-premise, Hybrid	Limited by storage	apprenda.com
Armor Complete	Virtual Machines	Private	Available by request	armor.com
AT&T	iPaaS, CDN	Public, On-premise	Available by request	synaptic.att.com
Atomsphere by Dell Boomi	iPaaS	Public	Available by request	boomi.com
AWS Elastic Beanstalk	Application Containers, aPaaS	Public	Limited by storage	aws.amazon.com
Baasbox	mBaaS	Public, Private on-premise	30 days	baasbox.com
Bluebox Cloud by IBM	Virtual Machines	Private, Hybrid	Available by request	blueboxcloud.com
Celigo	iPaaS	Public	Available by request	celigo.com
CenturyLink AppFog	App Containers, aPaaS, DBaaS	Public	7 days	centurylinkcloud.com
CenturyLink Cloud	Virtual Machines, App Containers	Public, Private by provider, Hybrid	30 days	centurylinkcloud.com
Clever Cloud	App Containers, aPaaS, DBaaS	Any	No free trial	clever-cloud.com
Cloud 66	iPaaS, Container Management, CI	On-premise	Available by request	cloud66.com
Cloud Elements	iPaaS	Public, Private, Hybrid	Available by request	cloud-elements.com
CloudBees Jenkins Platform	CI as a Service	Private, On-premise	Available by request	cloudbees.com

PRODUCT	CLASSIFICATION	PUBLIC / PRIVATE	FREE TRIAL	WEBSITE
cloudControl	App Containers, aPaaS, DBaaS	Public, Private on-premise	14 days	cloudcontrol.com
CloudOne	Virtual Machines	Private	Available by request	oncloudone.com
CloudScape by RISC Networks	Cloud Migration Management	Public	Available by request	riscnetworks.com
Data Cloud Surround by Dimension Data	Virtual Machines, App Containers	Public, Private	Available by request	dimensiondata.com
Datapipe	Virtual Machines	Private by provider	None	datapipe.com
DCHQ	aPaaS, Container Management	Public, Private	Free tier available	dchq.io
Digital Ocean	Virtual Machines	Public	None	digitalocean.com
DivvyCloud	Cloud Infrastructure Automation	Public, Private	Available by request	divvycloud.com
Docker Swarm	Container management	Any	Open Source	docker.com
EMC Cloudscaling Elastic Cloud	Virtual Machines	Public, Private by provider, Hybrid	None	cloudscaling.com
Engine Yard	App Containers, aPaaS	Public	500 hours	engineyard.com
FatFractal	Virtual Machines, App Containers, aPaaS, DBaaS, MBaaS	Public, Private on-premise, Hybrid	30 days	fatfractal.com
Flowgear	iPaaS	Public	14 days	flowgear.net
Fujitsu Cloud Infrastructure as a Service	Virtual Machines	Public, Private	Available by request	fujitsu.com
Fujitsu Cloud Platform as a Service	iPaaS	Public, Private, On-premise	Available by request	fujitsu.com
G Platform by Indra Gnubila	aPaaS, DBaaS, iPaaS	Public	Available by request	gnubila.com
Google App Engine	App Containers, aPaaS	Public	None	cloud.google.com
Google Compute Engine	Virtual Machines	Public	None	cloud.google.com
Heroku	App Containers, aPaaS, DBaaS	Public	Limited by storage and 8 instances	heroku.com
Hosting.com	Virtual Machines	Any	None	hosting.com
HP Helion Public Cloud	Virtual Machines, iPaaS, DBaaS	Public	Limted by storage	hp.com
IBM Bluemix	Virtual Machines, App Containers, aPaaS, iPaaS, DBaaS, MBaaS	Public, Hybrid	30 days	ibm.com/software/bluemix/
IBM Softlayer	Virtual Machines	Any	30 days	ibm.com
Informatica	iPaaS	Public, Hybrid, On-premise	30 days	informatica.com
Interlok by Adaptris	iPaaS	Public	Available by request	adaptris.com

PRODUCT	CLASSIFICATION	PUBLIC / PRIVATE	FREE TRIAL	WEBSITE
Internap	Virtual Machines	Public, Private by provider, Hybrid	None	internap.com
Interoute	Virtual Machines	Public	Available by request	interoute.com
IronWorker by Iron.io	aPaaS, Container Management, iPaaS	Private	Available by request	iron.io
ITapp by ServiceNow	Cloud Infrastructure Management	Public, Private	14 days	itapp.com
Jelastic	Virtual Machines, App Containers, aPaaS, iPaaS	Any	Two weeks with hosting partner	jelastic.com
Jitterbit	iPaaS, Cloud Migration	Public, Hybrid, On-premise	30 days	jitterbit.com
Joyent	Virtual Machines	Public, Private on-premise, Hybrid	Free tier available	joyent.com
Kii	mBaaS	Any	Free tier available	kii.com
Kinvey	mBaaS	Private by-provider, Private on-premise	Limited by storage	kinvey.com
Kony MobileFabric	aPaaS, iPaaS, mBaaS	Public, Private on-premise, Hybrid	90 days	kony.com
Krystallize Technologies	Cloud Performance Management	Public	Available by request	krystallize.com
Kubernetes	Container management	Any	Open Source	kubernetes.io
Kumulos	mBaaS	Private by-provider	Free until app is deployed	kumulos.com
Linode	Virtual Machines	Public, Private by provider	None	linode.com
Lunacloud	Virtual Machines, App Containers, iPaaS, DBaaS, Model-Driven PaaS	Public	None	lunacloud.com
Mendix App Platform	aPaaS, DBaaS, Model-Driven PaaS	Any	Free tier available	mendix.com
Microsoft Azure	Virtual Machines, App Containers, aPaaS	Public	30 days	azure.microsoft.com
MIOedge by MIOsoft	aPaaS, Analytics-as-a-Service, iPaaS	Public, Private	Available by request	miosoft.com
Modulus by Progress Software	aPaaS, Container Management, Cloud Infrastructure Management	Public	Available by request	modulus.io
NaviSIte	Virtual Machines	Public, Private on-premise	None	navisite.com
New Relic APM	Application Performance Monitoring	Public	14 days	newrelic.com
Nginx Plus	Virtual Machines	Public, On-premise	30 days	ngnix.com
NTT Communications	Virtual Machines	Private	Available by request	ntt.com
OpenStack	Virtual Machines	Public, Private on-premise	Open Source	openstack.com
Oracle Cloud	aPaaS, DBaaS, iPaaS, App Containers, Virtual Machines	Any	Available by request	cloud.oracle.com

PRODUCT	CLASSIFICATION	PUBLIC / PRIVATE	FREE TRIAL	WEBSITE
OrangeScape Visual PaaS	aPaaS	Public, Private, On-premise	Available by request	orangescape.com
Outsystems Platform	aPaaS, iPaaS, DBaaS, MBaaS, Model-Driven PaaS	Any	30 days	outsystems.com
Peak 10	Virtual Machines	Public, Private	Available by request	peak10.com
Pivotal CF	App Containers, aPaaS, DBaaS	Public, Private on-premise	90 days	pivotal.com
Profitbricks	Virtual Machines	Public, Private by provider	14 days	profitbricks.com
Rackspace Open Cloud	Virtual Machines, DBaaS	Any	Free tier available	rackspace.com
Red Hat Enterprise Linux Openstack	Virtual Machines, App Containers, DBaaS	Public, Private on-premise	90 days	redhat.com
Red Hat JBoss Enterprise Application Platform for xPaaS	App Containers, aPaaS	Any	Open Source	openshift.com
Rollbase by Progress Software	aPaaS, Model-Driven PaaS	Any	30 days	telerik.com
Salesforce1	App Containers, aPaaS, iPaaS, DBaaS, Model-Driven PaaS	Public	Limited by storage	salesforce.com
SAP HANA	Virtual Machines, App Containers, aPaaS, iPaaS, DBaaS, MBaaS	Public	30 days	sap.com
SingleHop	Virtual Machines	Public, Hybrid	Available by request	singlehop.com
SkyOnDemand by TerraSky	iPaaS	Public, Hybrid, On-Premise	Available by request	terrasky.com
SnapLogic	iPaaS	Public, Hybrid, On-Premise	Available by request	snaplogic.com
Sungard	Virtual Machines	Public, Private by provider	None	sungard-as.com
Verizon Terremark	Virtual Machines	Any	None	verizon.com
Virtustream Cloud IaaS	Virtual Machines	Private by provider, Hybrid	None	virtustream.com
vSphere by VMware	Virtual Machines, Cloud Infrastructure Management	Hybrid	Available by request	vmware.com
webMethods Cloud by Software AG	aPaaS, iPaaS	Private, On-Premise	Available by request	softwareag.com
Windstream	Virtual Machines	Public, Private on-premise, Hybrid	None	windstream.com
WorkXpress PaaS	aPaaS, DBaaS, MBaaS	Public, Private on-premise, Private by-provider	30 days	workxpress.com
WSO2 App Cloud	Virtual Machines, App Containers, aPaaS, DBaaS, MBaaS	Public, Private on-premise, Hybrid	Free tier available	wso2.com
Youredi	iPaaS	Public	Available by request	youredi.com
Zayo	Virtual Machines	Public, Private, On-premise	30 days	zayo.com

GLOSSARY

APPLICATION CONTAINER
A lighter-weight, more granular resource isolation than a virtual machine (no need for hypervisor, kernel shared across containers, isolation within userspace).

AMAZON WEB SERVICES (AWS)
A huge suite (>50 by count) of *aaS provided by Amazon; many services highly elastic; availability regions distributed globally; largest public cloud provider by far.

BASE (BASIC AVAILABILITY, SOFT STATE, EVENTUAL CONSISTENCY)
An approach to storage that divides physical or virtual storage medium into independently addressable chunks ('blocks'); increases performance by narrowing search space (specified as a path) for a particular store or retrieve operation; often accessed via logical abstraction layer that adds metadata (filesystem, DBMS).

CLOUD BROKER
(Like any other broker) abstracts away from provider details to offer users easier access to cloud computing resources; often provides simplified API and/or human UI, data lifecycle management, and focused service integrations and aggregations.

CLOUDBURSTING
When an application that normally runs in a private datacenter adds additional public cloud resources in response to demand.

CONTENT DELIVERY NETWORK (CDN)
Physically distributed servers that provide (often static) content along paths optimized per user; decrease transport time and overall network load; simplify per-machine resource management; prevent DoS by distributing (and thereby absorbing) requests.

CLOUD COMPUTING (FOLLOWING NIST)
Ubiquitous, convenient, on-demand access to shared computing resources; offers on-demand self-service (without human interaction), broad network access, resource pooling (dynamically assigned as workloads vary), location independence (to varying degrees), rapid elasticity, metered service (charging only for resources used); generally offered at three fundamental service levels (IaaS, PaaS, SaaS); deployed for use within an organization (private cloud), for any organization or individual (public cloud), or some combination (hybrid cloud).

DEVOPS
The breaking of siloes between development and operations; the co-incentivization of features and uptime; involves automating as much of the release pipeline as possible, converging code branches and deployment environments wherever practical, providing transparency and feedback from production to all stakeholders, rapid deployment and recovery from failures, continuous delivery, etc.

HIGH AVAILABILITY
Highly available systems are fully or acceptably operational most of the time, where the approach of 'most' to 100% is often characterized by 'number of nines' (90% = one nine, 99.999% = five nines, etc.); requires redundancy (with clean handoffs), automatic failure detection (and often self-repair), and usually extensive testing and simulation.

INFRASTRUCTURE-AS-A-SERVICE (IAAS)
Basic computing resources (virtual servers, containers, storage, transport, database, caching, networking) offered according to the cloud computing paradigm (see definition above).

LOAD BALANCER
Distributes inbound requests across a server cluster, often (but not necessarily) via round-robin (request 1 is routed to server 1, request 2 to server 2, etc.).

MICROSERVICE ARCHITECTURE
Describes applications built as collection of single-process services communicating over constrained and easily managed channels (often HTTP), where each service does one well-defined business level task or set of tasks and scales independently of other services. Microservice component boundaries map onto bounded contexts in Domain-Driven Design. The aim is to make changes easier, deployment faster, technology<->business match tighter, infrastructure more automated, conceptual and data models more flexible, and applications more resilient to failure.

MODEL-DRIVEN PAAS
A rapid application delivery (RAD) platform in the cloud; aimed at low-code business users and/or developers who need to build new enterprise applications quickly; may allow deeper access from code (e.g. Salesforce Lightning -> Force.com).

MULTI-TENANCY
Describes running one software instance for each client/user, where users are isolated by metadata within the application architecture. Ideal for some kinds of many-user SaaS (e.g. Force.com); less ideal where deeper schema flexibility and resource isolation are required.

OBJECT STORAGE
An approach to storage that marks chunks of data ('object') with application- or use-case-specific metadata and a unique identifier (key). More loosely coupled with physical medium and more tightly coupled with object use than block storage.

SERVICE LEVEL AGREEMENT (SLA)
A contract that quantitatively specifies resources offered, with availability (often uptime and response time); see 'high availability' for one common type of cloud SLA.

SOFTWARE-DEFINED NETWORKING (SDN)
An approach to networking that separates data and control. In non-software-defined networks, control (generation of routing tables) and data (packet forwarding) are both located in routers. In software-defined networks, control across routers is abstracted into a central controller, which can determine routing programmatically across the entire network.