What to Know When Working on Cloud-Based Apps

By **Tom Smith**, dzone.com
March 8th, 2016

To gather insights for *DZone's Cloud Research Guide*, scheduled for release in April, 2016, we spoke to 28 executives, from 23 companies, who develop and deploy applications in the cloud for their own company or for their clients.

Here's who we talked 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 | Faisal Memon, Product Manager, NGINX | Arvind Mehrotra, President and Global Business Head – Infrastructure Management Services, NIIT Technologies | Jens Eckels, Director, PaaS Business Group, Oracle | Pat Harper, SVP Operations, PGi | Joan Wrabetz, CTO, Quali | Partha Seetala, CTO, Robin Systems | Nick Kephart, Senior Director Product Marketing, ThousandEyes | Kiran Bondalapati, CTO and Co-Founder, ZeroStack.

When we asked these executives "What do developers need to keep in mind when working with applications for the cloud?" here's what they told us:

- Application performance management should be proactive versus reactive, especially when there’s a lot going on in the development phase versus the

production phase. You need to **get more information in test during the development stage**. APM tools will help you test the app before reaching production, and you can't afford to wait until production if you want to get to market as quickly as possible.

- **Understand 12-factor applications**. How to consume services remote to my applications? Understanding dependencies is primary.
- Developers should **be cloud agnostic**. All of their applications should be portable. For higher level applications, need to blueprint regardless of the cloud. When writing an app, be cloud agnostic, build generic APIs, encapsulate cloud-specific dependencies. You cannot assume a certain VM. Pick a region, it's important for security but a V center doesn't have regions because they're built for private clouds. This is something to keep in mind at the blueprint stage. The biggest challenge is how to abstract for cloud specifics. If you're building for the public cloud, build to scale out with microservices-multiple influences of a service to scale. The sandbox is a microcosm with Amazon and V container interface that enables you to see the conflicts between the two.
- Think outside of the box because old developer methodologies no longer are effective. Understand the business needs and the needs of the customer and provide a simple solution to the problem. **Tailor the methodology to the needs.** Have a consistent vision and shared knowledge about the objective and the problem to be solved (use UTrack from JetBrains and JIRA to collaborate, would like to be more socially-based). You have to solve complexity by adding simplicity.
- Just do it. **Decide on a simple app and get on AWS and build it.** That's why AWS provides access. APIs and tutorials are an easy and great way to learn cloud service.
- Think about when apps scale and what parts of the app need to scale. Architect in containers with shared apps like logins. **Be able to scale as needed**. Think through usage patterns and if they should be the same for different types of users. Understand what pieces you don't want to build-know what backends as a service you might want to use (i.e. API gateway, data management service). Know what's ok to be locked into the cloud provider versus what's giving you differentiation in the market.
- **Learn logic, take math, study art** to understand the user interface—get experience with more progressive programming languages. Leverage your skills to transform

apps. Find a mentor. Build a strong foundation of knowledge and you never know what the outcome will be. Follow your passion.

- **Leverage hackathons for fast, rapid app development.** Think about security. Encourage developers and software teams to participate in hackathons to generate faster applications and better UX. Fail fast before deployed rather than using a testing organization.
- Think about the parts that can fail–use AWS and GCQ to bring up instances to see what works and what fails. **Continually test and see what efficiencies you're getting.**
- Do not make assumptions of where apps are running. **Build apps that can run in any environment.** Build in security from the beginning. Build abstractions to move from one cloud to the other. Don't get tied to a single cloud or technology.
- **Security** - easily the most important aspect, especially isolating each tenant's data and protecting sensitive data. **Scalability** - must be able to handle potential large spikes in usage during certain times of the day or month. **Cost Control** - Optimizing the application for cost effectiveness is critical since public cloud costs can quickly add up during attempts to optimize performance and scalability. The infrastructure costs are now barred by the software company and not the customer. **Logging** - Important to be able to debug issues and capture everything about a failure so it can be fixed with little assistance from the customer. **Deployability** - A primary benefit of SaaS is being able to deploy quickly and at any time. The architecture must be able to handle live deployments that cause zero downtime and are invisible to the customer. **Multi-tenancy** - the toughest hurdle for all on-premise developers since the application is not being built for a single customer behind a firewall. **Automation** - the underlying infrastructure must be automated to facilitate deployment and scalability without a large manpower footprint.
- Know how to design patterns with special books. Design patterns in the cloud are not well-documented or understood. Think about patterns up front. Abstract design patterns ensure you're relevant and ensure your career will grow. When you know basic patterns, you can figure out complex patterns. Don't repeat the mistakes of the past. More people are beginning to document patterns. **Observe what people are doing, find people with experience, read blogs and online resources.** Many companies are openly talking about their cloud architecture.

Look at how 10 different companies are doing it and look for the patterns. This will provide more content to consider when you are building solutions.

- Do not forget the cloud is a very powerful place for development. Leverage it along with everything it offers. **Push the cloud to lead with regards to fast iteration, frequent deployments, and quality apps.**
- Scalability, security. Devs need to **think about security from the beginning.** Have a zero downtime mentality.
- **Performance and instrumentation.** Build as much instrumentation into the code as possible to make troubleshooting easier in case there's a problem. It can cost millions of dollars to be down for a couple of hours. Know APM configuration and build instrumentation. Identify what are and are not acceptable performance levels.
- 1) Tech perspective of V5 of no app. Server as an architecture is where we're going (i.e. Amazon Lambda-have a piece of code and when you call it, it will run [Nano services]). 2) A lot of platforms are becoming higher in the stack with business functionality to become better at creating predictive models, defining business algorithms that feed the models. **Look at domain knowledge and combine that with abstract thinking.** Google Tensor Flow, Google Now, and Microsoft are now providing open source machine learning. Think about how to solve problems.
- Keep security top of mind as more and more apps get hacked. **Security and reliability go hand in hand.** Ensure the app can scale to handle spikes in traffic and long-term growth. Get value to the customer ASAP.
- Know the tools you have at your disposal. **Design for scale and performance.** Anticipate change. Foresee what the product is doing and how to integrate with other products being used by businesses. Make your apps extensible. Put security in up front or your app will not be compliant. Put UX at the center of design.
- **Automation and validation.** If you use modern tooling within cloud applications and it doesn't fit with what you're trying to do you may be using an old model with a new product. Look on the internet and see what others are doing. A large majority of migrations are using open source tooling (e.g. Netflix). Look and you'll find good ideas and solutions.
- **Start with security leaving no flaws to exploit.** Always use a multi-tenant environment to scale cost efficiently.

- Developers and DevOps become cloud natives understanding the application but not the service delivery. Lots of tooling a built-in functionality but not everything you'll need so there's still a lot to learn regarding the performance, UX, and how the app is served up to the end user. It's valuable to know all of these things, to know if your app is usable and user-friendly. Know dependencies and scalability. **Talk with DevOps and operations to understand how your code will be served.**

What suggestions would you give to developers working on cloud-based apps?