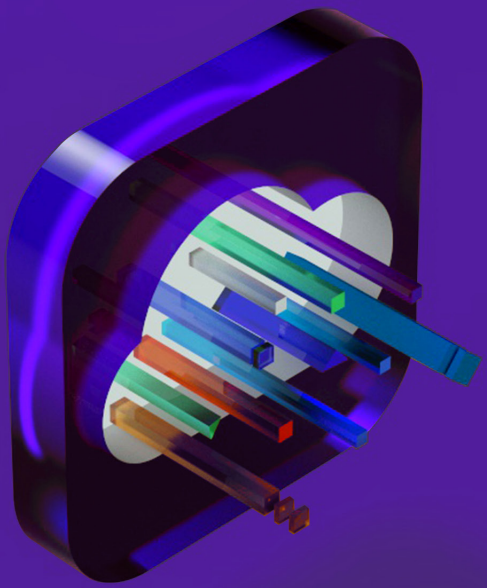
Kong

EBOOK

Is Your New App a Candidate for Cloud Native Development?

A checklist for building your first cloud native application



Content

Even in our day, few people understand what is necessary to properly design and build a cloud application architecture that exploits all of the amenities in a modern cloud services environment. The result of this lamentable lack of skills and experience is the deployment of many poorly functioning cloud applications. In comparison with mature, reliable legacy monolithic applications, such poorly built cloud applications don't deliver the value that enterprises have come to expect.

Why cloud native?

As architects and engineers have sought to exploit cloud computing, the essential drivers of cloud native architecture have become the following:

Although the Cloud has its benefits, for many large organizations, it's difficult to lift and shift all workloads completely to the Cloud. This difficulty results in multiple different environments to maintain. Since security is easier to enforce with visibility and consistency across systems, the hybrid model poses a challenge since consistency is not always achievable.

- **Resilience** — It's risky to assume that your deployment environments and the networks are permanent. They will eventually change — and substantially so. Your critical applications may not receive any warning to accommodate a smooth shutdown. Therefore, it's vital that you design for failure and assume that some services on which you depend could disappear at any time.

- **Discovery** — Services that support your applications must be readily locatable and accessible by other services. Services are often built to scale dynamically and change locations. Therefore, software must have the ability to readily discover the locations of other components and services—and communicate with them.

- **Scalability** — Though you may manage your applications in the cloud, these apps won't achieve the efficiencies of a cloud native architecture if they are unable to scale horizontally.

This article presents an outline of the four stages of cloud native application development: assessment, planning, deployment and verification. To be successful, it's important that your team thoroughly consider all of the items in each of the checklists below.

Consider cloud native for your next application

We know what you're thinking: a cloud native project is never truly done. As with legacy applications, it will be necessary to extend and integrate your cloud native app. Whenever you engage new services to solve business problems with cloud native solutions, this cycle will need to begin again. Of course, you may need to employ other technologies to meet different objectives. So, while this is an expansive guide, we must stress the importance of customizing it to your specific context.

As you consider building your first cloud native application, the first thing on your checklist should be to ensure that cloud native is the appropriate design paradigm. Simplicity and effectiveness should always be the primary goal of architectural design. It's important, also, to remember that conventional application development approaches don't apply to cloud native projects.

Don’t build another monolith

The monolith still lives on, everywhere. From this point onward, resolve not to introduce any mainframe or monolithic rigidity into your cloud native application.

Yes, we may chuckle at the old-school companies that continue to milk their old mainframe contracts for all they can. The picture in our minds is one of a sterile, industrial room with massive air conditioning that envelops one massive computer. It’s tended by an army of middle-aged, stiffly dressed engineers that still use pencils and old calculators.

There is considerable value in many of these legacy systems, but these systems are monolithic and inflexible. Most don’t scale well, are fraught with duplication and excess redundancy, limit creativity, and lack extensibility that compares with cloud computing platforms.

There is a tendency, in many teams, to build a mainframe each time a new application development effort is begun. Eventually, a rigid mainframe-like iron framing is seen to appear, no matter how small the application starts. This is because such teams unthinkingly add new features, bolt on inflexible interfaces, and use duct tape and shoestring to strap various trinkets and baubles onto a once-nimble. Sadly, too many monolithic applications are really a virtual mainframe — an opaque box of functionality that is all too commonly found among many business systems.

Why do many teams still build monoliths?

There are two major reasons for the tendency to build monoliths:

The dominant reason is, ironically, a desire for efficiency, which always seeks to simplify complex arrays of things by categorization and also by expanding the reach of entities that seem to work well. For example, if you like your house, then you might add a garage. Analogously, if you like your customer database, then you might extend it by adding columns indefinitely. In a simple architecture, it seems best to go on bolting things to well-working things. There are so many IT examples out there that have shown most clearly that this soon leads to a sticky ball of mud.

Another reason is that ideal software development practices entail code reuse. However, this is a rare achievement in the first release of a product. The reality for most teams is that code is added willy-nilly. Many hacks and riggings are put in place until the app becomes unbearably fragile and virtually untestable. Object-oriented approaches often devolve into a model that is functional only because of an authoritarian god-object that sits in the middle of it all. This, dear reader, is a mini-monolith.

Is your app a candidate for cloud native?

Cloud native architectures mitigate — or eliminate — many of the challenges with monolithic applications. As you prepare to build a cloud native application, it's important to remember an elemental fact: a monolith will never work with the new paradigms that you must adopt and the new techniquest that you must implement to successfully build and deploy a cloud native application.

Because microservices architecture is complex, only specific workflows and some application types are suitable for development with microservices. The best candidates for cloud native applications include large, complex systems that consist of many dynamic or evolving subsystems. So, it's vital that you take the time to assess whether your app is a good candidate for a cloud native architecture.

Assessment

If you don't already realize it, there's no cloud service that will accommodate every type of cloud native architecture. Dazzled by marketing or reputation, many teams will gravitate toward a preferable cloud environment without taking the time to carefully consider how well that cloud services provider aligns with the application architecture — and also the culture and maturity of the organization.

As you evaluate a cloud service provider, stop and ask some foundational questions. Consider the answer to these questions from both the perspective of the development team and the cloud service providers (CSPs) that you evaluate.

- **Is your app — and your team — a good fit for this CSP, that one or another?** Though there are countless CSPs, many of them are unlikely to accommodate your specific requirements. Some CSPs may offer expansive, instantaneous availability of virtual servers. This is fantastic if you're among the organizations that need such a capability. Another CSP could deliver comprehensive managed services that might be incompatible or excessively irrelevant to your DevOps context. Don't make the mistake of casually choosing a touted leader of a specific cloud domain, or you may inadvertently select a provider that will have a negative impact on your success. Carefully seek a CSP that provides business value that well exceeds purely technical features, such as easing compliance issues, help with managing risk and an excellent shared-responsibility model.

- **What are the capacity requirements for your app?** As your team considers the workload that you'll need to manage, a few different CSPs may rise up in your ranking. This will depend on whether the application workload will require hundreds of lightweight, ephemeral servers/containers or need a smaller pool of compute resources that will need to expand slowly over time. Consider the overall landscape over which your application will extend and ensure that your chosen CSP can support the upper limits of compute, network and storage capacity that you expect to need in the future.

- **Are you counting all of the costs?** The price of cloud services often exceeds the rough-estimate numbers that you'll encounter in most price lists. To better size up your actual costs, take time to carefully model the actual scenarios that you expect to encounter. Also, look for special charges that will apply for geographic locations, bandwidth consumption, API calls, backup storage and other incidentals. Remember to consider the additional costs of support plans, onboarding, performance testing and staging environments.

- **How will your existing tools work in cloud native development and deployment?** Likely, your team already has an existing toolset and set of processes. If it won't be easy to adapt these tools to the CSP that you choose, it's important for the team to make time to learn the tools that the CSP supports.

Planning

After your team chooses the cloud service provider(s) that will support your app, you'll need to design the application and also plan how you build and deploy that application.

Here are the most critical things that you should consider when planning out the design and deployment of your cloud native application.

- **How will your existing tools, licenses, and compliance requirements center on microservices?** The most central concern in the design of your cloud native app is that

it should be a microservices architecture. Because microservices architecture is complex, only specific workflows and some application types are suitable for development with microservices. The best candidates for cloud native applications include large, complex systems that consist of many dynamic or evolving subsystems. Because of the huge upside, it's worth taking the time to learn the tools and explore the solutions. The additional effort to design a microservice-driven, cloud native application is justifiable by realizing the benefits of longer-term agility, flexibility, higher efficiency and easier application maintenance. For simple scenarios, it is often better to employ a monolithic design or simpler n-tier architecture.

- **Carefully design the topology of the architecture.** A cloud native architecture is different from anything that you have on-premise. Data services, cloud servers and cloud networks function quite differently than their on-premise equivalents, so it's important to carefully analyze the capabilities of your CSP. Then, it may be necessary to rethink your architecture and processes for better alignment with the new landscape.

- **Will you encounter performance bottlenecks?** If your application will need various integration points with external or on-premise systems that will not be migrating to the cloud environment, then you're likely to encounter performance problems that arise from external connections. Cloud services are available in varying sizes and shapes, so be sure to provide your applications with sufficient computing resources to handle the workload that you expect.

- **By what means will users access the application?** It would be quite frustrating to surprise the user community with an entirely different set of credentials for accessing an array of cloud services. Consider how you can employ the single sign-on (SSO) mechanisms that most CSPs offer and ensure that your team deals with this as a priority design issue.

- **How will your team deploy code, data and configuration updates to the environment?** Check carefully for compatibility between the cloud services that you select and your existing system administration tools. If you're using modern configuration management tools, there may be extensions available that work well with your CSP. Take time to thoroughly simulate the deployment and configuration process to identify anything that needs integration or improvement.

- **What business processes might need to change to exploit the cloud native application?** Your company procedures for on-premise requisitioning hardware, handling change management, testing and deployment probably won't work with the CSP. Any attempt to map these processes onto the agile, automated, self-service cloud environment may result in the loss of the key benefits that your team was seeking in the pursuit of a cloud native application. Perform a thorough assessment of the process changes that will be necessary.

- **How will this application operate after deployment?** What will happen when a change request arises? What monitoring will take place, and what indicators will be used? How

will your team troubleshoot problems? Take a close look at the entire lifecycle of the cloud workload and catalog all of the maintenance tasks.

- **How will you train project staff?** It's important for your team to become very familiar with all of the new cloud services. You can work with your CSP to get training and prepare onboarding materials that will help all staff members get comfortable with the cloud services environment — including architects, developers, project managers and DevOps personnel.

- **Will you conduct a pilot that alleviates all of the concerns above?** It's a setup for failure if you proceed with only simple prototypes. Carefully plan the actual installation and configuration of the application in the cloud environment. Then, perform a succession of trial runs. Develop solid expertise by acquiring full familiarity with all of the interfaces, constraints and capabilities.

Deployment

With careful planning, deployment to the cloud should be rather uneventful. If you can answer these questions below with confidence, then your team will be well-equipped to deal with any surprises that occur.

- **How do you plan to distribute the application and its data to the cloud?** There are a number of approaches to migrate your application and data to the CSP. For medium-size workloads, use simple copy procedures or commands to pass data over an Internet connection. For large sets of data, you stand to incur lengthy transfer tasks and significant CSP bandwidth charges. Consider compressing the data and copying it to a temporary staging environment in the target cloud prior to final transfer — or physically ship data disks to your CSP (if they offer support for this).

- **What security and access controls will you establish?** During development and testing, you may use staging servers and temporary storage repositories. Take care to ensure that you think through all access and data security issues. This is especially critical for any sensitive data such as passwords, PII or PHI.

- **How will you migrate your data?** It may be necessary for your team to move data to a self-managed database or a database-as-a-service environment. Take inventory of all the tools that the CSP provides and what limitations you'll encounter with respect to data structures or large volumes of data.

- **Plan to create a new set of cloud environment metadata.** It would be nice to achieve cloud portability and avoid vendor lock-in. We all need to face the fact that this isn't proving to be feasible. While it's true that VMs and a lot of code can be portable, the environment metadata is quite specific to each CSP. User accounts, policies, permissions, load balancers and such will vary from one CSP to the next. Be sure that you take care to properly configure all aspects of the environment that will host your cloud native application and its data.

- **Make solid preparations if you need to move virtual machines to the cloud.** Building an app locally and then moving one or more VMs is another way to transfer an application to a CSP. Proceed very carefully if you go this route. Keep in mind that you may encounter some configuration issues if the VM was previously attached to an on-premise network. Also consider the domain and the types of disk that were in use. This lift-and-shift approach may appear simple and easy, but it's typically more complex than anyone expects the first time.

Testing and Verification

After completing the deployment of your cloud native application, it's critical that your team verify that all aspects of the application perform exactly as you and your users expect.

- **Is the application reachable?** Begin by verifying that all application services are accessible and provide basic functionality. Users at each level should be able to access their respective functions, all microservices should be operational and responsive, and each internal component should be communicating without any errors.

- **Verify that all necessary administrative tools can access the cloud environment.** As mentioned above, it's important to do this in the planning stage. Now your team should verify that everything works now that the application has been deployed. All management tools must have a clear view into the CSP environment and have full capability to monitor the application.

- **Has all the data successfully transferred to the cloud?** With automated methods and some spot-checking, you should be able to verify that all metadata, reference data and transaction data has successfully transferred — or it's accessible by the application from the cloud environment.

Summary

Building a cloud native application architecture requires that you focus on a number of new things. And yet, many conventional concepts remain important. This includes planning, sound design, careful configuration and development, thorough testing, and a readiness to learn from missteps. The best development teams that deploy applications on public or private cloud platforms will invariably make mistakes. However, they are successful in large part because they stand ready to recognize, correct and learn. Such teams refine their techniques and collaborate better so that they carve out the most effective path to cloud native development.

As you consider moving ahead with your cloud native development project, it's vital to reach agreement that microservice orientation is essential — and must be given top priority. This is important to accept even though it will likely require longer (initial) development lifecycles and somewhat larger budgets. Though it costs more to build a cloud native application in comparison with a conventional application, the rewards can be quite substantial. For many applications, it's a smart investment.

Research

1. https://read.acloud.guru/if-you-cant-strangle-the-monolith-starve-it-to-death-fcc824d3c82
2. https://docs.microsoft.com/en-us/dotnet/standard/modernize-with-azure-and-containers/modernize-existing-apps-to-cloud-optimized/what-about-cloud-native-applications
3. https://medium.com/@maxy_ermayank/cloud-native-architecture-application-architecture-17510923a439
4. https://medium.com/techietalks/why-you-should-go-cloud-native-546616c1cdda
5. http://adriangrigoras.com/blog/architecture-review-checklist/

Konghq.com

Kong Inc.
contact@konghq.com

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