Logi ANALYTICS

BEST PRACTICES FOR DEPLOYING & SCALING EMBEDDED ANALYTICS

Like every other feature in an application, embedded dashboards and reports are constantly evolving. Analytics features that set applications apart five years ago—like data visualizations and interactive reports—are now considered the bare minimum. Even once-modern capabilities like embedded self-service analytics are now commonplace.

Today, sophisticated capabilities such as adaptive security, predictive analytics, workflow, and write-back are taking analytics far beyond basic dashboards and reports. With cutting-edge capabilities like these, application teams are able to drive revenue and differentiate their products from the competition.

As embedded analytics becomes increasingly complex, deploying and scaling it gets more complicated. What should development, IT, and DevOps teams keep in mind when embedding analytics in their applications?

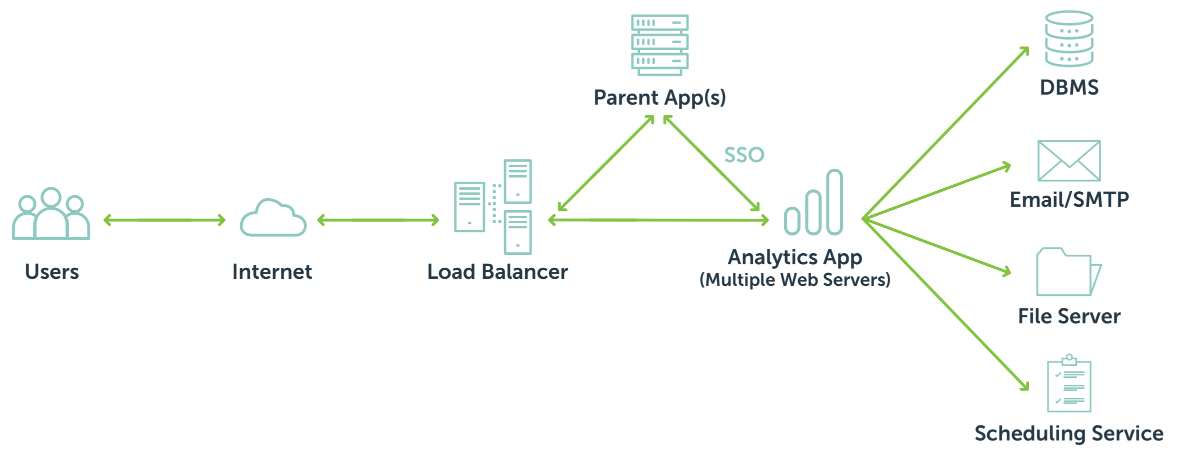
This ebook explains how to simplify the deployment and scalability of your embedded analytics, along with important considerations for your environment architecture, application design, and deployment.

Contents:

Environmental Architecture

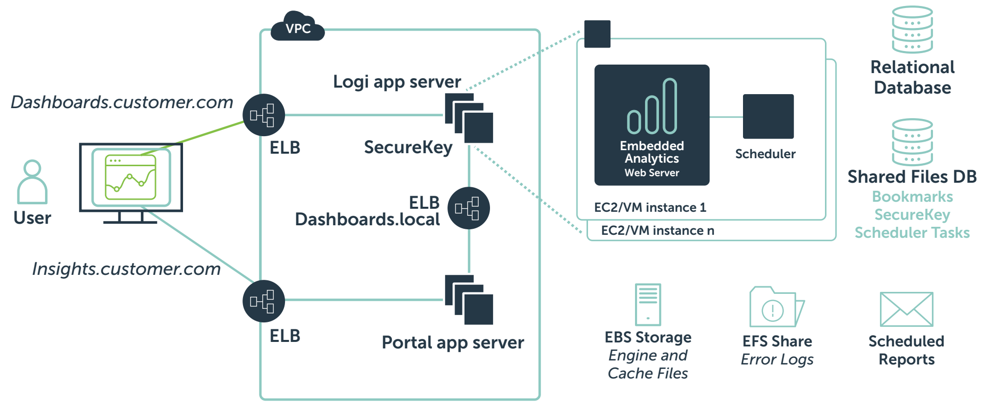
A typical web architecture is hosted behind a load balancer. The web server communicates with the database to retrieve data and present it to the user.

An embedded analytics architecture is very similar, since it's also dealing with a web application that happens to be communicating with another parent application. But for a successful embedded analytics application, a sound database with schema designed for analytics will better serve your end users. Some of the strategies for an analytics-oriented database schema include flattened and aggregated tables.



You can see an example of a typical cloud deployment below. Servers are deployed to multiple Availability Zones with load balancers set to target them—ensuring they can carry out the additional features of embedded analytics without negatively impacting application performance, as well as accessing the data needed to supply the right analytics to customers.

Typical cloud environment



Application Design

The next step is to consider how to package and host your analytics application. Traditionally, the application is hosted on a physical server or a virtual machine (VM). However, some application teams are now migrating to containers. There's a lot of buzz around how containers can streamline your deployments. Both options—VMs and containers—are still relevant. Your decision should be based on your company's specific needs.

	Physical Servers / Virtual Machines (VMs)	Containers
PROS	✓ Traditional, robust model ✓ Flexible delivering multiple services	✓ Lightweight ✓ Simple to deploy and scale
CONS	X Environment Dependent X Slower process to generate and maintain	X Shared resource constraints

Containers are particularly useful for microservice-oriented architectures with distributable services. They're also good for prototyping and can be replicated relatively quickly to meet demand. However, their file storage is not persistent and requires external tools to support.

Physical servers and virtual machines have been tried and tested for at least 20 years, so you can package multiple services and capabilities into a virtual machine and still deliver in a load balance environment. VMs are good at supporting monolithic application architectures as well as handling multiple functionalities.

VMs do have some drawbacks. They are environment dependent. You cannot have a virtual machine that's hosted in, say, Windows, and then switch over to a Linux container. Their turnaround time is also slightly longer than containers.

Nevertheless, the flexibility of the services that can be generalized through VMs is good for establishing embedded analytics. Depending on your capabilities, you can choose either a VM or a container-based approach.

Deployment

In a typical environment for embedded analytics, the traditional deployment model has involved a delivery mechanism. You might host the mechanism, or you might provide it as a service to your end customers. Though on-premises hosting is the best option for delivering mission-critical services, the ease of deployment and maintenance with cloud hosting makes it the better option for embedded analytics.

What are the benefits and drawbacks of hosting on premises or in the cloud?

	On Premises	Cloud
PROS	✓ Lower cost ✓ Full control over all aspects of hosting	✓ Lightweight ✓ Simple to deploy and scale ✓ Latest technology stack
CONS	X Capacity must be pre-determined X High availability is limited X Maintenance is more difficult X Technology upgrades can be expensive	X Scaling can get expensive X Potentially tied to one vendor

The main benefit of **hosting your application** within your own environment or colo data center is it costs less. Because you are in charge of your architecture, you can decide how to procure and scale analytics. Plus, you can maintain complete control over how you deliver your services.

There are some drawbacks of hosting on premises, however. The turnaround time to procure and deliver on hardware is longer, and the hardware itself is more difficult and expensive to upgrade. Crucially, scaling is less straightforward with self-hosting than it is when in the cloud. You will have to pre-determine the size of your environment, or risk being limited in the availability of your application.

Hosting in the cloud makes scaling considerably easier. All you have to do is script it out, request a virtual machine (or docker containers), and within a matter of minutes your environment has scaled to cope with additional traffic. While the cloud often has a low cost of entry, as you grow and scale your costs may increase and you'll become more heavily reliant on your vendor. But the cloud solves many of the drawbacks of the on-premises model. It's lower maintenance, as the vendor takes care of the hardware. Upgrading technology is easier, and doesn't come with the risk of downtime. And, you can scale dynamically.

Because it makes scaling easy, cloud hosting is recommended—as long as you keep an eye on expenses.

Packaging and Deployment Workflow

Your embedded analytics solution should work seamlessly with your current application. When built as a web application, embedded analytics platforms fit well into modern code management and deployment situations. Deep integration means your development team can leverage their existing tools and processes.

A combination of code management and scripted deployment processes helps add value by extending the analytics features in your application. The best way to enhance features is with a phased approach.

Here, we outline a typical phasing strategy used by many application teams:

Phase 1

1. Start with developer-defined, managed dashboard content.
2. Provide self-service features with limited sets of data. Limits could include certain scopes (for example, expose detail-level data to users within constraints of a specific territory, or choose a category and sub-category combination within a limited timeframe like a quarter or a year) or higher-order aggregates (such as at the state level vs. zip code level).
3. Monitor how users interact with data and compare with how it performs.

Phase 2

1. Review database and application performance and fine-tune systems by improving reporting data models, adding database indexes, increasing application server resources, and so on.
2. Expose additional datasets and more granular dashboards with interactive drilldowns based on user feedback from prior phase.
3. Monitor for database and application performance.

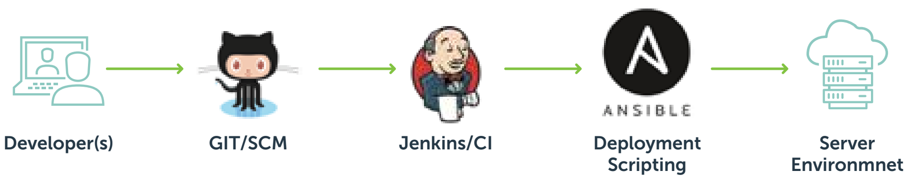
Phase Next

1. Iterate over prior phase.
2. Review enhancements provided by the embedded analytics platform over newer versions.
3. Add or remove features based on user feedback.

Read more about data environments in the white paper: Toward a Modern Data Architecture for Embedded Analytics

How can this phasing strategy be tied to code versioning and deployment workflows?

Consider this typical workflow:



Development and/or DevOps configures the embedded analytics solution in a development environment, either using the integrated development environment (IDE) provided by the embedded analytics vendor or a web interface. Multiple developers can collaborate on the same effort. Configuration definitions (and code) are committed to a source code management system. Once feature development is complete, the code is tagged for release. DevOps captures a tagged version of the software, packages it, and deploys to a web server environment. DevOps can use scripting tools to automate packaging software, procure server resources, and deploy software to servers.

Code Versioning and Packaging an Embedded Analytics Application

Most development shops use a code versioning system to maintain their application source code. Common tools include GIT, (Microsoft) TFS, and SVN. Your embedded analytics platform should be able to leverage these tools to give developers a similar level of code management and code sharing capabilities.

An embedded analytics solution includes an engine and a developer-defined configuration which drives the engine to generate content for end users. Configuration definitions are typically maintained in XML or JSON files.

You have two options in versioning code:

1. Commit configuration definitions in a source code management system, then package engine files separately and handle directly in the deployment step.
2. Commit a complete solution, including configuration definitions and engine, to a source code management (SCM) system.

If you maintain configuration definitions separate from engine files, you'll see a few benefits—including the ability to leverage partial deployments for smaller, incremental changes. This is due to a smaller set of files needing to be updated and deployed.

However, since modern source code management systems are able to support larger file structures, maintaining a complete solution in the SCM allows developers to streamline the delivery of product enhancements. It is recommended to maintain your complete embedded analytics solution in your SCM.

Deployment

Most software shops nowadays maintain some form of scripting to simplify the deployment of applications to server environments. Common tools include Jenkins, Chef, Ansible, Puppet, TFS, and Octopus. These tools provide two main benefits:

1. Script or automate compiling and/or packaging of software and prepare for deployment
2. Script or automate setup of environment and deploy software to environment

Integrating embedded analytics software into your deployment workflow can be seamless. You can tie it with your existing application deployment cycle or set it up on a separate workflow. The level of integration depends on how tightly your current application's features are coupled with your embedded analytics solution. Most application teams tend to maintain a decoupled solution, which allows for delivering enhancements to embedded dashboards and reports at a different pace from rest of your application suite.

Defining a sound workflow for code management and deployment will allow your product team to quickly identify feature requests from customers, develop and test changes, and deliver enhancements to customers in a flexible, phased cycle.

Manual	→ Scripted	→ CI/CD
• Define Feature development and delivery lifecycle that suits your environment	• Implement scripts to automate packaging and deploying application	• Build cadence to deliver features regularly and often
• Test manual deployment of application	• Consider Configuration updates separate from Engine upgrades	
• Review application performance and adjust web server capacity and environment configurations		

Conclusion

Embedding analytics in your application doesn't have to be a one-step undertaking. In fact, rolling out features gradually is beneficial because it allows you to progressively improve your application. You can get new capabilities out the door quickly, test them with customers, and constantly innovate.

By using an embedded analytics platform, you can pick and choose what you deliver to customers and add new capabilities over time. Choose a platform that integrates with industry-standard products and services (such as cloud environments) and is built to fit into your complex environment and infrastructure.

Logi Analytics is the only developer grade embedded analytics platform on the market: Watch a Demo Today

About Logi Analytics