Building Microservices for Speed and Agility

A practical approach to creating your next-generation application architecture with CA Live API Creator

Speed and agility matter

Businesses are being disrupted every day by digital upstarts that find ways to address new market requirements before the more established companies can respond.

Despite talented IT teams and years of head start in both architectural and development work, it is still difficult to respond to these challenges using traditional development patterns centered around monolithic software applications. It's simply impossible to get to market quickly when applications need to be maintained, modified and scaled as a single entity by a large, heavily inter-dependent team.

From this need has arisen the microservices paradigm: a set of patterns for software architecture, development, deployment and culture that focus on speed and agility. From small, independent services and teams to automated deployment to fault tolerance and resiliency, these patterns help accelerate time to market.

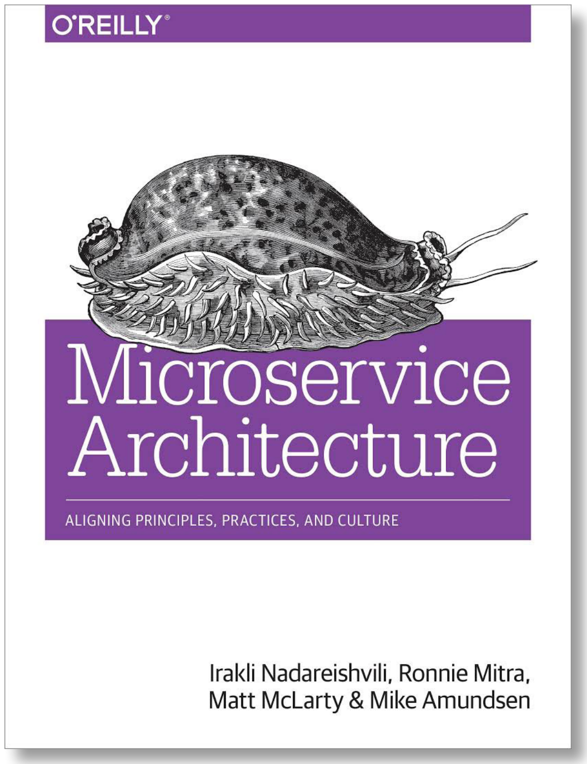
Enterprises that can harness the value of microservices have the potential to digitally transform their business, address the application economy, proactively disrupt their own business and thrive, rather than waiting for a competitor to corner the market.

From the thought leaders

The API Academy is a team of industry experts from CA Technologies that travel the globe speaking, teaching and consulting on best practices for APIs, microservices and related technologies. Its recent book, Microservice Architecture, provides a valuable primer on the subject, including these two fundamental definitions:

"A microservice is an independently deployable component of bounded scope that supports interoperability through message-based communication. Microservice architecture is a style of engineering highly automated, evolvable software systems made up of capability-aligned microservices."

Let's explore some of the main concepts around microservices and how they might be addressed in an enterprise environment.


O'REILLY
Microservice Architecture
ALIGNING PRINCIPLES, PRACTICES, AND CULTURE
Irakli Nadareishvili, Ronnie Mitra, Matt McLarty & Mike Amundsen

Get your complimentary copy from ca.com

Delving further into the Microservice Architecture book, we discover some primary characteristics of microservices. Microservices are:

- Small in size
- Messaging-enabled
- Bounded by contexts
- Autonomously developed
- Independently deployable
- Decentralized
- Built and released with automated processes

Micro by nature, macro by value

There is debate in the industry over the precise definition of these terms—does small refer to the granularity of the service, the size of the team building it or the physical size of the deployed code? Less controversial is the idea that these characteristics can translate into increased velocity, resilience and scalability. Expressed another way, the goal of these architectures should be "speed and safety at scale."

While these benefits are good for the software development experience within an organization, they also help realize tangible benefits for the business. Quicker time to market enables new business models and more effective offerings. Stronger resilience and higher availability ensure a good customer experience. And improved scalability allows individual components to grow or shrink with the business, saving additional hosting and maintenance costs.

In other words, this new architectural style offers some direct and some indirect benefits to the business's bottom line.

Building a microservice with bounded contexts

Whether a business is adopting a microservices architecture in order to break down an existing monolith or as a method for building new applications from scratch, the concept of designing microservices around bounded contexts is an important one. Any given microservice should be at a level of granularity such that it can be independently developed and deployed, maintaining the functionality needed for standalone operation while at the same time drawing clear boundaries between it and other components of the architecture.

It's often a time-consuming and iterative analysis process just to define the bounded context, and to identify which requirements should be capabilities of a given microservice and which should be broken out into a separate context. For each bounded context (service) in a new microservices development scenario, there should be a self-contained bundle that provides everything needed for independent operation. These embedded dependencies include:

- A messaging layer that defines how to interact with the service (usually an API)
- A logic layer that defines how the data available in this context will be processed to fit a need
- An integration/data layer that provides access to the portion of data relevant to this service

In practice, each of these layers could also potentially require a significant amount of development effort, especially in an enterprise setting where there are expectations around the feature set at each layer.

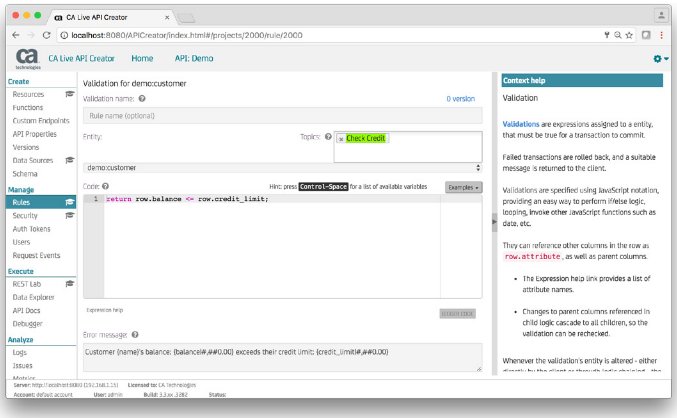
- First of all, the service needs to be consumable through an API of some kind. This should be user-friendly and robust, allowing for real-world patterns around pagination, searching, sorting, documentation, etc. These capabilities can be difficult and repetitive to implement for each new service.
- The logic layer should codify what happens when the API is called to execute some operation or update some data, relieving the calling application from needing awareness of (or compensation for) side effects or consequences of that operation. Understanding the ramifications of any potential incoming request and the order of internal operations needed to account for those changes is a painful, error-prone process that can become difficult to maintain as services evolve.
- The integration/data layer should provide access to all of the data needed within this context, in a vocabulary specific to this context. This means building an object-relational mapping, incorporating SQL/NoSQL object stores relevant to this context, or defining the logic and code necessary to make external calls to other microservices.

In some cases, the tasks related to building microservices are difficult enough that whole patterns have been built around accounting for the complexity.

- The Event Sourcing pattern manages the current state of a system by keeping an ongoing record of every transaction that has made a state change, essentially working with a “snapshot” of the current system but maintaining a long-term audit log of every update. This pattern, while not simple to implement either, is an alternative to managing the real-time dependencies of updates in create, read, update and delete (CRUD) based models.
- Similarly, the Command Query Responsibility Segregation (CQRS) pattern separates interactions with a system into distinct data-querying and data-update operations, each with a context-specific interface and vocabulary. This avoids having to retain the same degree of interface complexity for calls in different contexts but requires additional development effort to build two separate representations of the same data set.

Support for these (and other) patterns and characteristics provides a basis for requirements for any architecture enabling microservices development, deployment and management.

An exciting new option for microservices development—CA Live API Creator



Validation for demo:customer

Validation name: 0 version

Rule name (optional)

Entity: demo:customer

Topics: Check Credit

Code: Hint: press Control-Space for a list of available variables

```
return row.balance <= row.credit_limit;
```

Error message: Customer {name}'s balance: {balance|#,##0.00} exceeds their credit limit: {credit_limit|#,##0.00}

Validation

Validations are expressions assigned to a entity, that must be true for a transaction to commit.

Failed transactions are rolled back, and a suitable message is returned to the client.

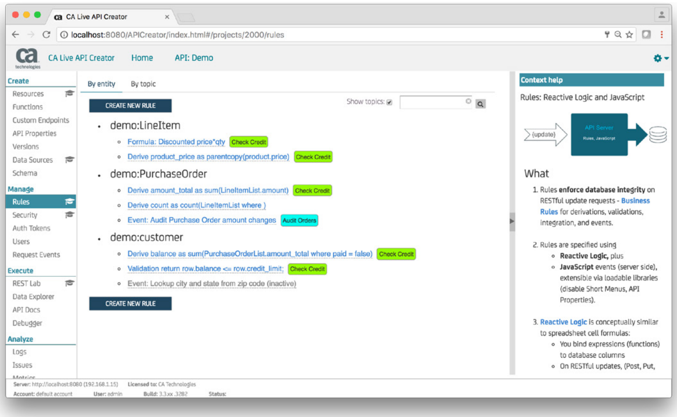
Validations are specified using JavaScript notation, providing an easy way to perform if/else logic, looping, invoke other JavaScript functions such as date, etc.

They can reference other columns in the row as `row.attribute`, as well as parent columns.

- The Expression help link provides a list of attribute names.
- Changes to parent columns referenced in child logic cascade to all children, so the validation can be rechecked.

Whenever the validation's entity is altered - either

Server: http://localhost:8080 (192.168.1.15) Licensed to: CA Technologies
Account: default account User: admin Build: 3.3.xx .3282 Status:



- demo:LineItem
  - Formula: Discounted price*qty — Check Credit
  - Derive product_price as parentcopy(product.price) — Check Credit
- demo:PurchaseOrder
  - Derive amount_total as sum(LineItemList.amount) — Check Credit
  - Derive count as count(LineItemList where)
  - Event: Audit Purchase Order amount changes — Audit Orders
- demo:customer
  - Derive balance as sum(PurchaseOrderList.amount_total where paid = false) — Check Credit
  - Validation return row.balance <= row.credit_limit; — Check Credit
  - Event: Lookup city and state from zip code (inactive)

Rules: Reactive Logic and JavaScript

What

1. Rules **enforce database integrity** on RESTful update requests - Business Rules for derivations, validations, integration, and events.
2. Rules are specified using
   - **Reactive Logic,** plus
   - **JavaScript** events (server side), extensible via loadable libraries (disable Short Menus, API Properties).
3. Reactive Logic is conceptually similar to spreadsheet cell formulas:
   - You bind expressions (functions) to database columns
   - On RESTful updates, (Post, Put,

Server: http://localhost:8080 (192.168.1.15) Licensed to: CA Technologies
Account: default account User: admin Build: 3.3.xx .3282 Status:

40x more concise

10x faster to market than code

CA Live API Creator is an innovative, low-code approach to developing, deploying and executing microservices and other application backends in a fraction of the time taken by traditional application development methods. CA Live API Creator's powerful and easy-to-use interfaces (UI/API/CLI), point-and-click resource definition, declarative business logic, built-in developer tools and interactive data integration and exploration capabilities make it an ideal choice for building independent and deployment-ready microservices.

CA Live API Creator supports both design-time definition and runtime execution of microservices from the ground up, providing all the layers—API, logic and integration/data—necessary for independent development and deployment in a microservice architecture. In particular, CA Live API Creator uses Reactive Logic to define business rules that are 40 times more concise than writing the same logic using traditional coding methods. This makes development of services much faster and their maintenance more manageable, thereby enabling two fundamental underpinnings of microservices: speed and agility.

In addition, CA Live API Creator easily supports common patterns seen in microservices architectures, including Event Sourcing and CQRS, without adding additional complexity to the environment. It provides full CRUD access to each defined resource but also allows functional APIs to operate on those resources or stand alone. Users can define their desired level of service granularity and runtime isolation, and CA Live API Creator will support it—whether the services are micro, mini or macro.

CA Live API Creator's power and flexibility deliver complete, running microservices 10 times faster than alternative approaches, with the safety and scale necessary to confidently drive your business forward.

CA Live API Creator offers a choice of development methodologies

Every microservices initiative should begin with an analysis of the current and desired architecture, processes and culture. Among other discoveries, this analysis will reveal what application monoliths can be broken down into microservices or which independent services can be created for new applications. The current state and preferred strategy of adoption can suggest different methodologies for microservice development. CA Live API Creator directly addresses various approaches, including those described below:

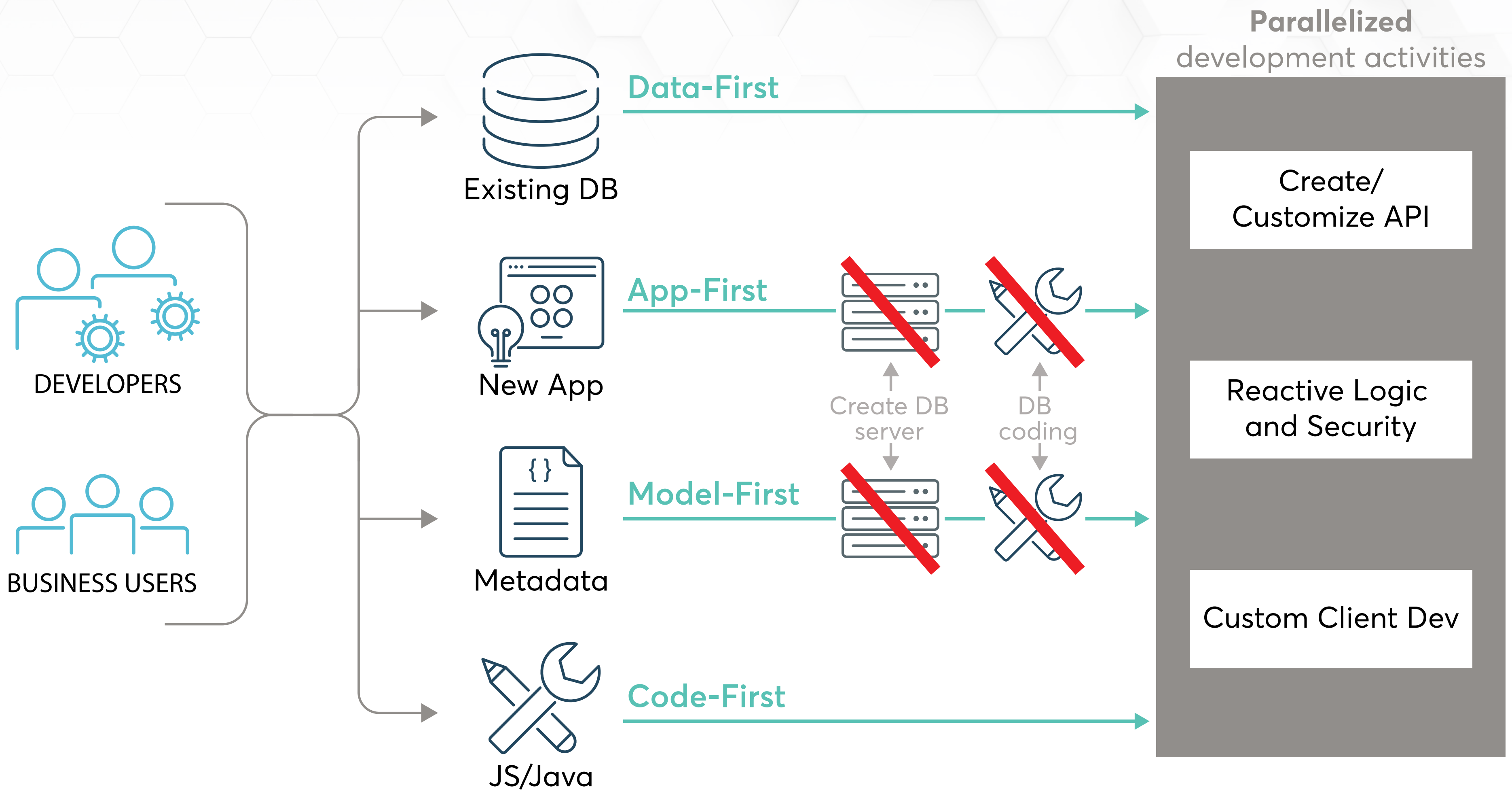
A data-first approach to microservices development is most suitable if there is a legacy data source and a new application is needed around it. For instance, you could decompose an existing monolith by extracting some independent portion of the underlying data and providing a microservices-driven data access layer, or you could create a novel data-as-a-service offering providing immediate value to siloes of existing data.

An app-first approach starts with an idea of a bounded context in which to build a microservice. In a point-and-click application-level UI, developers can design a new model with vocabulary and relationships customized to a specific domain. CA Live API Creator builds the database on the fly, constructs the data model based on the application designed, and automatically creates the APIs necessary to support it. This is a good top-down approach to building complete microservices (or just plain APIs) without requiring deep database skills.

A model-first approach helps transition from resource model metadata to a complete, running microservice exposed through APIs. CA Live API Creator builds the underlying database and resulting APIs from a simple, human-readable and -writeable JSON document; existing Swagger metadata can be converted as well. The whole process with CA Live API Creator is a couple of clicks or API/CLI calls away.

A code-first approach starts without a database, or builds very quickly on a simple database-backed framework, to write arbitrary microservice code. CA Live API Creator provides rich developer tooling, including a powerful JavaScript runtime engine, access to a full Java® stack, inline code editors with code completion, helper libraries to turn complicated tasks like message format transformation or API callouts into a single line operation, and an inline debugger that lets you set breakpoints and step through the code execution a line at a time, seeing the current value of variables as you go. For microservices implementing "function-as-a-service"–type scenarios, this is a fantastic approach.

None of these approaches need to be exclusive—once you begin and have the basic framework of APIs, logic and data in place, you can continue to iterate on your services at any level. Customize the resource definitions, enhance and extend the existing logic, build clients against the working APIs—and do it all in parallel to be able to quickly iterate and improve the design from all angles.



Deploying microservices is a snap with CA Live API Creator

While designing microservices in CA Live API Creator, developers get equivalent functionality across whatever interface they choose to interact with—UI, CLI or API. Everything that happens in the CA Live API Creator UI is also available for programmatic scripting and incorporation into deployment and operations toolchains. Export/import, source control integration and environment promotion across dev, test, prod, etc. are all scriptable, allowing for significant automation.

CA Live API Creator can run on a broad range of application servers and public and private cloud platforms, as well as Docker. This broad range of platforms guarantees maximum flexibility for running services created using CA Live API Creator.

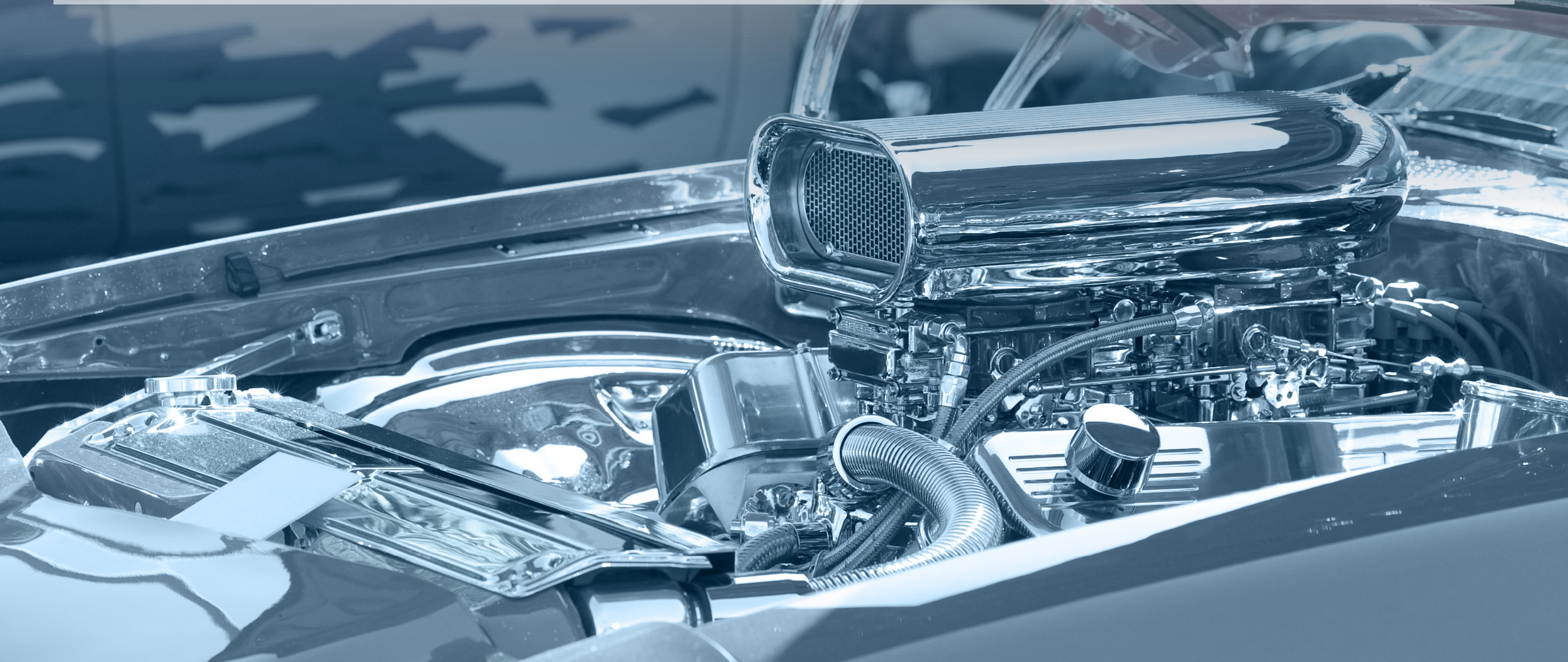
Supported application server platforms include: Oracle® WebLogic, IBM WebSphere®, Apache Tomcat, JBoss, Jetty and Glassfish. In addition, CA Live API Creator can be deployed on Microsoft Azure, Pivotal Cloud Foundry or Amazon Elastic Beanstalk, or consumed via the Amazon Web Services Marketplace—to be used in conjunction with other public/private cloud tools and technologies.

Perhaps most applicable to a microservices architecture, CA Live API Creator has been tuned for containerized deployments using Docker, running in either persistent or non-persistent mode and scaling up in a cluster with a single command. Whatever platforms are chosen for microservices deployment and operations, or for broader IT architectures, CA Live API Creator likely supports them.

Running a microservice in CA Live API Creator

Running a microservice in CA Live API Creator is no more complicated than designing it. The service is live and immediately ready to process traffic—not as a prototype, but as a robust service on a full-featured API server. Projects can be migrated manually or in a scripted fashion to production instances, which can be spun up on demand as individual nodes or as part of a cluster. These instances are lightweight and can be ephemeral, supporting the scale necessary for peak traffic but scaling down when that volume drops. CA Live API Creator can also support the Immutable Server pattern, being replaced instead of modified for server upgrades or project updates.

When running, CA Live API Creator is optimized on both the client side and server side for maximum performance. On the server side, database overhead is reduced by eliminating/minimizing SQL statements, in-transaction caching and careful lock management. Client side latency and chattiness is reduced using rich resources, pagination and optimistic locking. Security is managed via simple declarative rules that apply to a user or role across a broad set of endpoint options and/or row/column data access rights.



CA Live API Creator for microservices

Basic requirements (check). Real business value (bonus).

Let's revisit the microservices characteristics discussed earlier as a recap of CA Live API Creator capabilities.

- **Small in size:** CA Live API Creator enables microservices that are small when measured logically, physically or even in effort required. Developers are granted complete autonomy over logical service granularity, can deploy instances to anything from a Raspberry Pi to a massively scaled cluster and can accomplish an unprecedented amount of work with significantly less time and effort.
- **Messaging enabled:** Every microservice comes with a robust set of CRUD APIs for interacting with resources, as well as the ability to add functional APIs for more workflow-oriented use cases. The APIs are consistent and full-featured, providing pagination, sorting, searching, filtering, hypermedia links to related resources, multiple message formats and many parameters for customization.
- **Bounded by contexts:** Logical contexts are easily defined, iterated on and represented in the UI and the resulting APIs, including through custom API interfaces that abstract the service vocabulary from the underlying data source.
- **Autonomously developed:** Microservices developed in CA Live API Creator are independent and can stand alone or be composed into larger services or applications; a single developer or small team of developers can develop the API, business logic and data integration needed to be completely dependency-free.
- **Independently deployable:** Deployment and runtime isolation of microservices is completely independent of other systems and configurable by the developer. A single microservice can be tied to an entire server, or many can be deployed on a single server. This isolation flexibility is available at many levels: endpoints, resources, projects, accounts and servers.
- **Decentralized:** CA Live API Creator instances can live just about anywhere, in any architecture, including on-premises, public cloud, private cloud and platform as a service (PaaS). They can be dispersed geographically or by environment.
- **Built and released with automated processes:** Everything configured in the CA Live API Creator UI can also be done via CLI and/or API, so creation, imports, exports and migrations can all be entirely scripted.

As a microservices development and execution platform, CA Live API Creator provides more than just technical benefits—it also provides real business value. Accelerating the microservices lifecycle offers an opportunity to innovate by tackling a long backlog, addressing novel business models and "failing fast" when trying out new ideas that couldn't have been attempted using traditional processes. With less-error-prone and easier-to-maintain systems, the focus can shift to enhancing applications to improve customer experience, exploring new mobile or Internet of Things (IoT) scenarios, or disrupting new markets.

Move to Microservices and Accelerate Your Digital Transformation

You want to deliver new innovations, release apps faster and take advantage of new opportunities, but legacy applications and infrastructure are holding you back. Transition to a modern architecture by decomposing monolithic applications into agile microservices—independently created, managed and scaled. Your business will be able to act faster and developers will love the easy access to APIs that give them the freedom to focus on customer experience.

Start with Microservice Strategy and Design

CA has the API Academy, a team of API thought leaders who wrote the book on microservice architecture and provide organizations with the education and consulting they need to build better APIs and microservices, improve software delivery and execute on broader digital strategies.

Read **API Academy Microservice Best Practices**

Building Microservices

CA Live API Creator is the only automated, low-code microservices development solution and works up to 10 times faster than other approaches. It creates and exposes domain-driven microservices and REST APIs as application backends, providing access to orchestrated data and functionality from both new and legacy systems.

Learn more about **CA Live API Creator**

Orchestrate and Secure Microservices

Using an API gateway is the best-practice approach to ensuring your microservice architecture is secure and well-orchestrated. The industry-leading CA API Gateway is containerized and deployable in Docker®, which enables architects and developers to manage discovery, orchestration and transformation in a microservices environment, while providing best-in-class OAuth security and authentication to protect your business.

Learn more about **CA API Gateway**

Launch new microservices at the speed of light

Start building Microservices today with a trial of CA Live API Creator. Get started at **ca.com/createapis**.

Learn how CA Technologies can help you with your Microservice architecture. Visit **ca.com/microservices**.

CA Technologies (NASDAQ: CA) creates software that fuels transformation for companies and enables them to seize the opportunities of the application economy. Software is at the heart of every business, in every industry. From planning to development to management and security, CA is working with companies worldwide to change the way we live, transact, and communicate—across mobile, private, and public cloud, distributed and mainframe environments. Learn more at ca.com.

CS200-276793_0617

ca® technologies