ActiveState®

ACTIVEGO

FOR MICROSERVICES

SERVICE DISCOVERY
Supports Consul for service registration and discovery

DISTRIBUTED TRACING
Supports OpenTracing for debugging inter-service communications

CACHING
Support for local in-memory caching & distributed caching

USE LESS RESOURCES
Compiles to a small binary – no JVM/local interpreter required

NATIVE CONCURRENCY
Maximum performance on multi-core CPUs

REDUCE RISK
Security scanned; license reviewed & backed by SLA

COMPATIBILITY
100% compatible with community open source Go

Scale and Manage your Microservices in Production, backed by Commercial Support

Microservices are fast becoming a key, strategic initiative within the enterprise, allowing architects to better address the challenges of scaling applications, while supporting more flexible and agile development practices.

Simply put, microservices are small, loosely coupled, self-contained services that feature discrete functionality which communicate with each other over a network. By replacing monolithic applications or even SOA-based web services, microservices allow the enterprise to solve business problems faster by iterating multiple, small, non-dependent services quicker than versioning a single large application.

WHY GO?

Go was developed to address the cloud computing revolution when APIs were dominating and continuous integration/continuous deployment (CI/CD) was becoming commonplace. In other words, Go was built for the way enterprises develop applications today.

But Go's real claim to fame is native concurrency, which means that multiple tasks can be executed at the same time. While you could spend time and resources to implement concurrency in older programming languages like Java and C, Go offers it out of the box. For microservices-based applications, which are typically distributed across multiple, multi-core/multi-CPU servers, building them in Go maximizes application scalability and performance while minimizing development costs.

When it comes time to deploy, Go microservices are compiled into small, discrete binaries that do not depend on local interpreters or JVMs to run. In a fast-paced CI/CD world, where minimizing deployment overhead is critical to helping enterprises keep up with the pace of business change, Go microservices shine.

ACTIVEGO

FOR MICROSERVICES

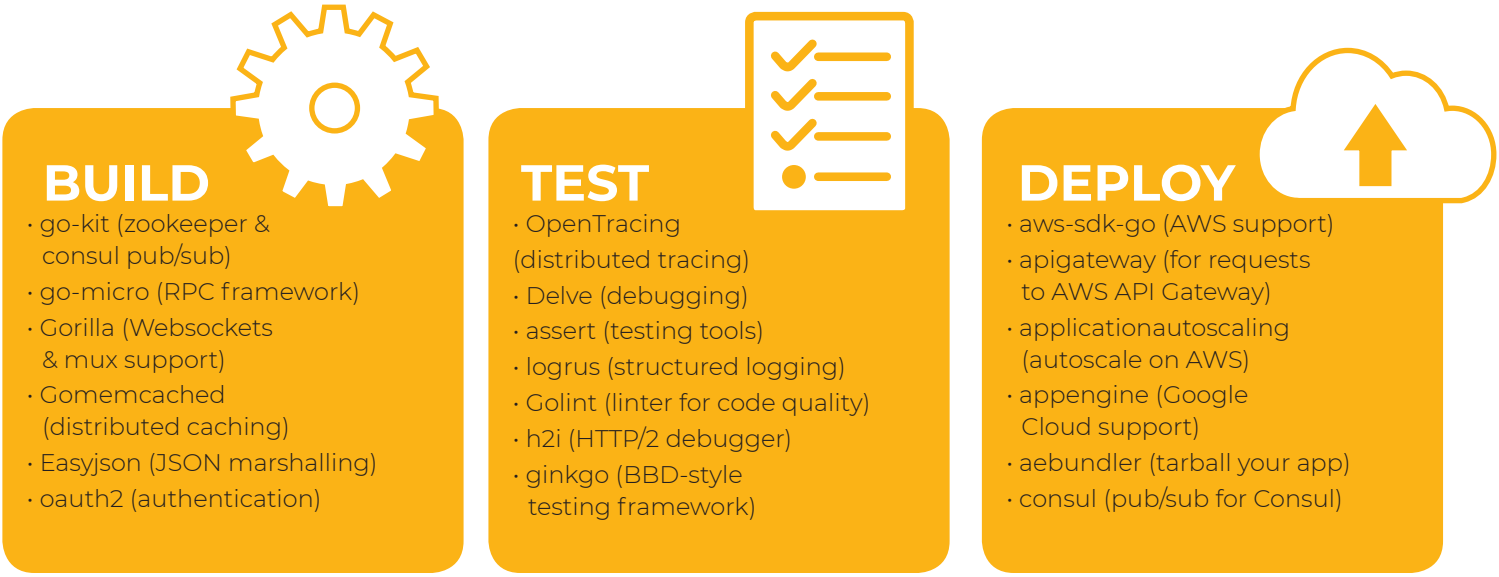

Figure 1: Some of Go's included libraries and third-party packages

Like all Go initiatives, building a microservice starts by defining a set of imported packages you want to leverage in your service. Figure 1 lists a set of commonly used Go community packages (as well as native tools) that you might want to consider to simplify the building of your service.

Communication between services needs to be fast and reliable, which is why Go includes a built-in net/http library; appropriate for REST-based endpoints that serve dynamic content. Endpoints that serve static content can benefit from libraries like fasthttp, which offer in-memory caching for smaller jobs, but there's always gomemcached for larger caching jobs.

Self-contained microservices typically include their own data layer. Go provides support for multiple relational (MySQL; Oracle, etc.) and non-relational databases (Cassandra, Mongodb, etc.), with streaming data support provided by sarama, a client library for Apache's Kafka.

Logging and error tracing are important for any application, but doubly so for applications that rely on distributed services. Consider using the included logrus for per service logging. To visualize the journey that transactions take across your networked services Go provides support for the OpenTracing project (distributed tracing) as well as zipkin (visualization).

Finally, many enterprises are choosing to deploy their Go microservices on public and private cloud infrastructures using Docker containers managed by Docker Swarm or Kubernetes for ease of orchestration, scaling and management.

While open source Go provides all of the tools and libraries discussed here, high-value staff can end up wasting days on the low-value work of installing and configuring packages before they are able to start writing code. ActiveGo not only comes pre-compiled with the most popular open source packages, but is also pre-optimized for compatibility and speed, ensuring microservices development teams can be productive right out of the box so you can get your microservices-based applications into production faster.

ActiveState®
website: www.activestate.com
Toll-free in NA: 1.866.631.4581
email: solutions@activestate.com

© 2018 ActiveState Software Inc. All rights reserved. ActiveState®, ActivePerl®, ActiveTcl®, ActivePython®, Komodo®, ActiveGo™, ActiveRuby™, ActiveNode™, ActiveLua™ and The Open Source Languages Company™ are all trademarks of ActiveState.