SPRING SUMMIT

开放共享 原生共融

北京 | 2017年8月26日

Pivotal

SERVERLESS ARCHITECTURE

Will Chen

2017.8.26

Agenda

- What is serverless?
- What’ s different?
- What is the benefits?
- What is the drawback?
- When to use serverless?
- Pivotal’ s approach for serverless

What is serverless

Backend as a services (BaaS)

- Serverless was first used to describe applications that significantly or fully depend on **3rd party applications / services (‘in the cloud’)** to manage server-side logic and state.

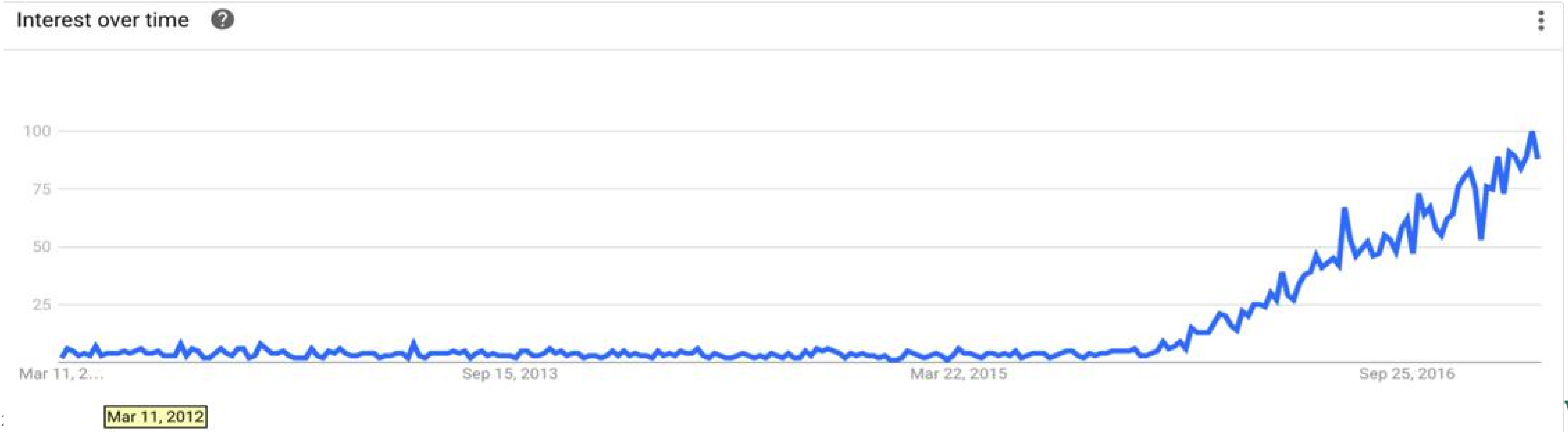
Function as a service(FaaS)

- Serverless can also mean applications where some amount of server-side logic is still written by the application developer but unlike traditional architectures is run in **stateless compute containers** that are **event-triggered**, ephemeral (may only last for one invocation), and fully managed by a 3rd party.

Pivotal

History of Serverless

- 2012 - Used to describe BaaS and Continuous Integration services run by third parties
- 2014 - AWS launched Lambda
- 2015 - AWS launched API gateway
- 2015 - AWS re:Invent The Serverless Company Using AWS Lambda
- 2016 - Serverless Conference

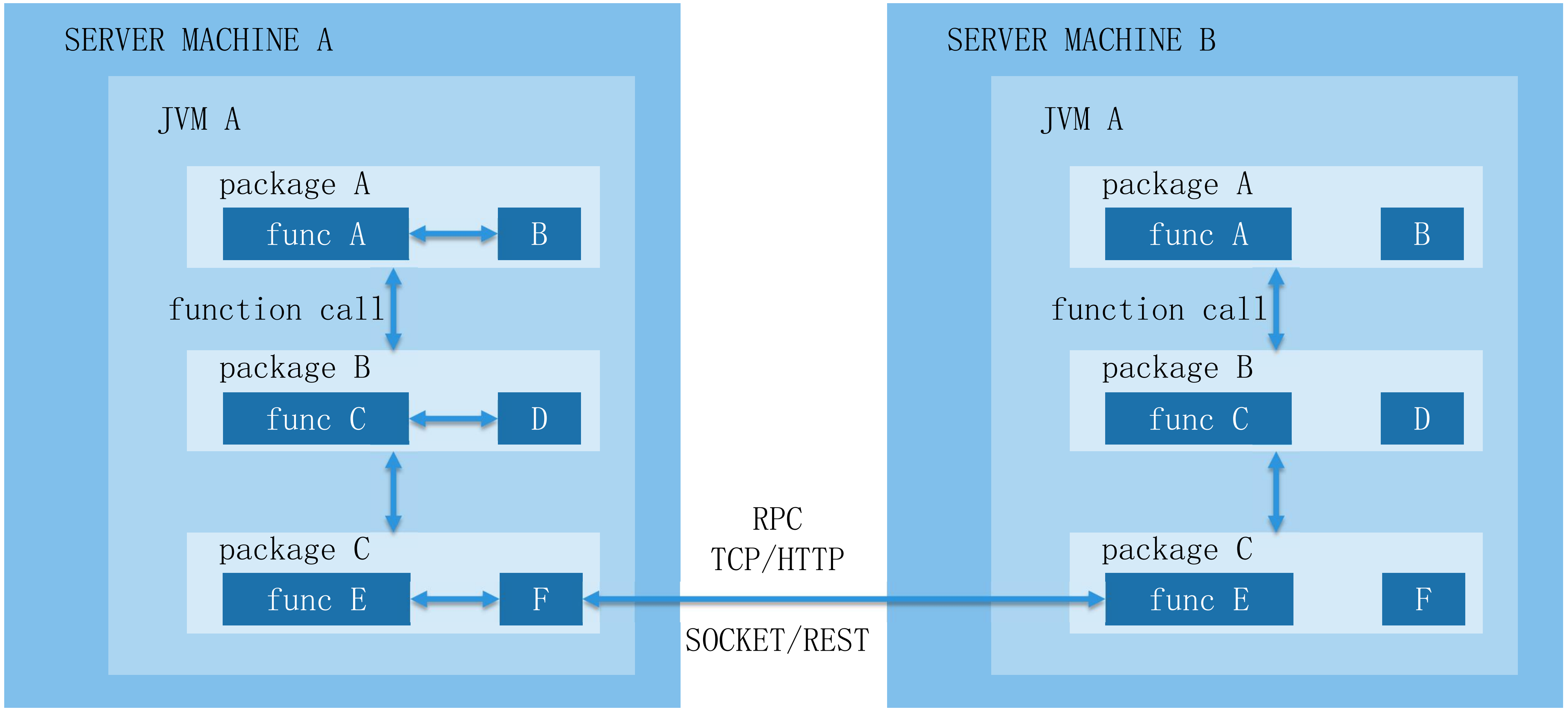


What’ s different

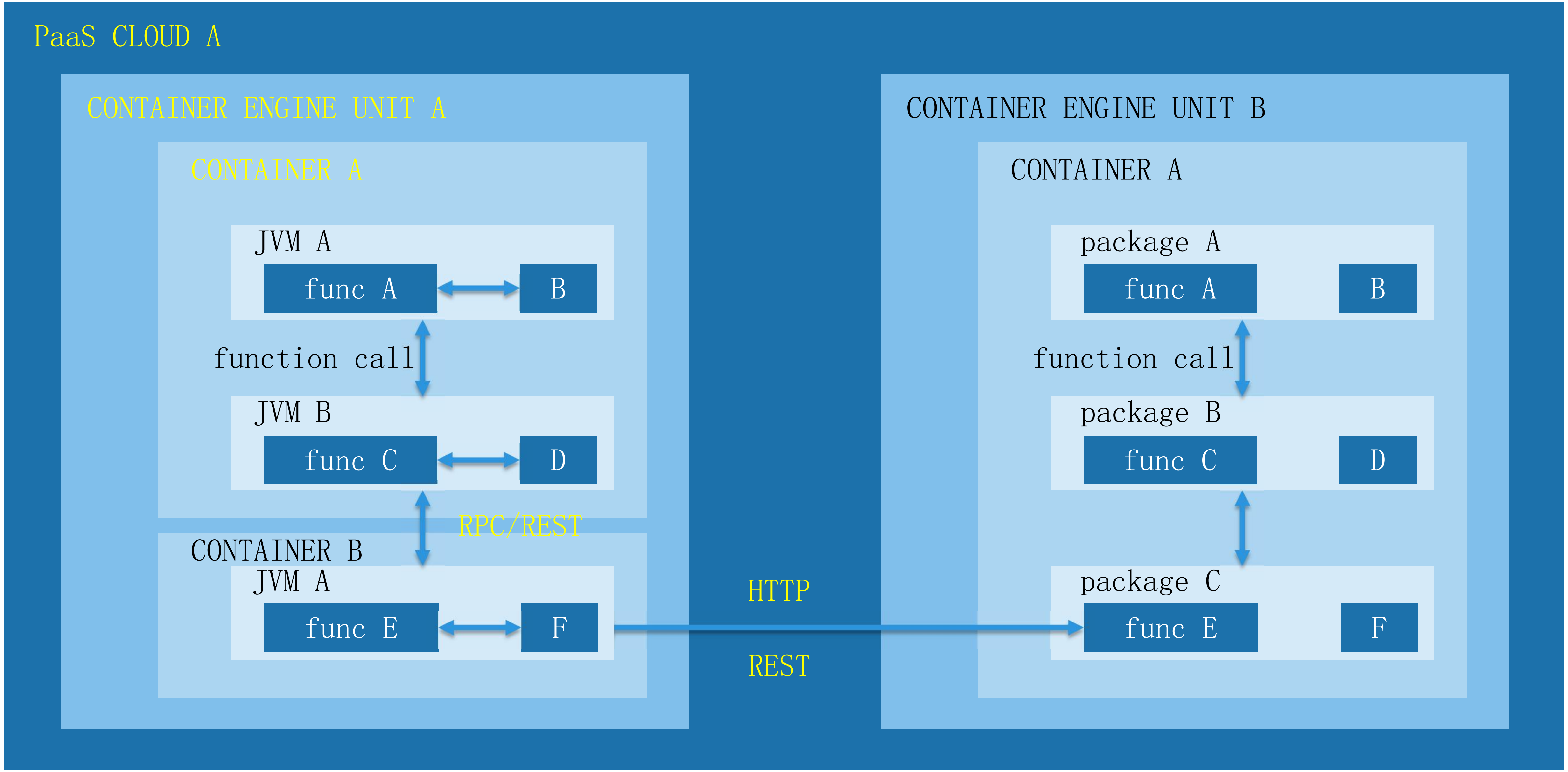
Serverless vs PaaS vs Container

- Granularity
  - Function vs Service
  - f->f.map(s->s.toString().toUpperCase())
- Management/goverenance
  - NoOps(no knowledge of middleware) vs DevOps
- Application Pattern
  - event centric vs non enforced

Monolith RPC

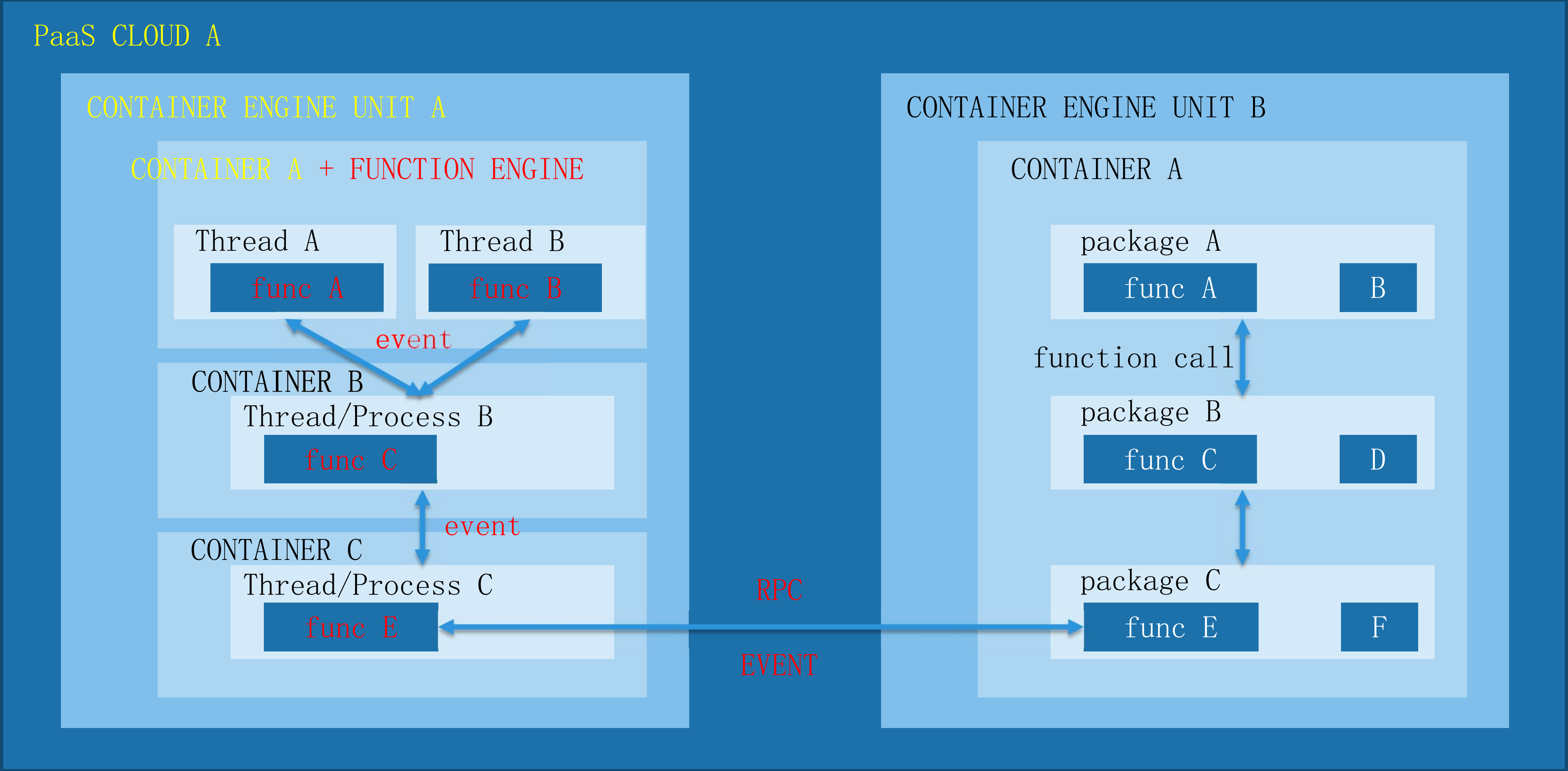


Microservice



Pivotal

Serverless


PaaS CLOUD A
CONTAINER ENGINE UNIT A
CONTAINER A + FUNCTION ENGINE
Thread A
func A
Thread B
func B
event
CONTAINER B
Thread/Process B
func C
event
CONTAINER C
Thread/Process C
func E
RPC
EVENT
CONTAINER ENGINE UNIT B
CONTAINER A
package A
func A
B
function call
package B
func C
D
package C
func E
F

From three different perspective

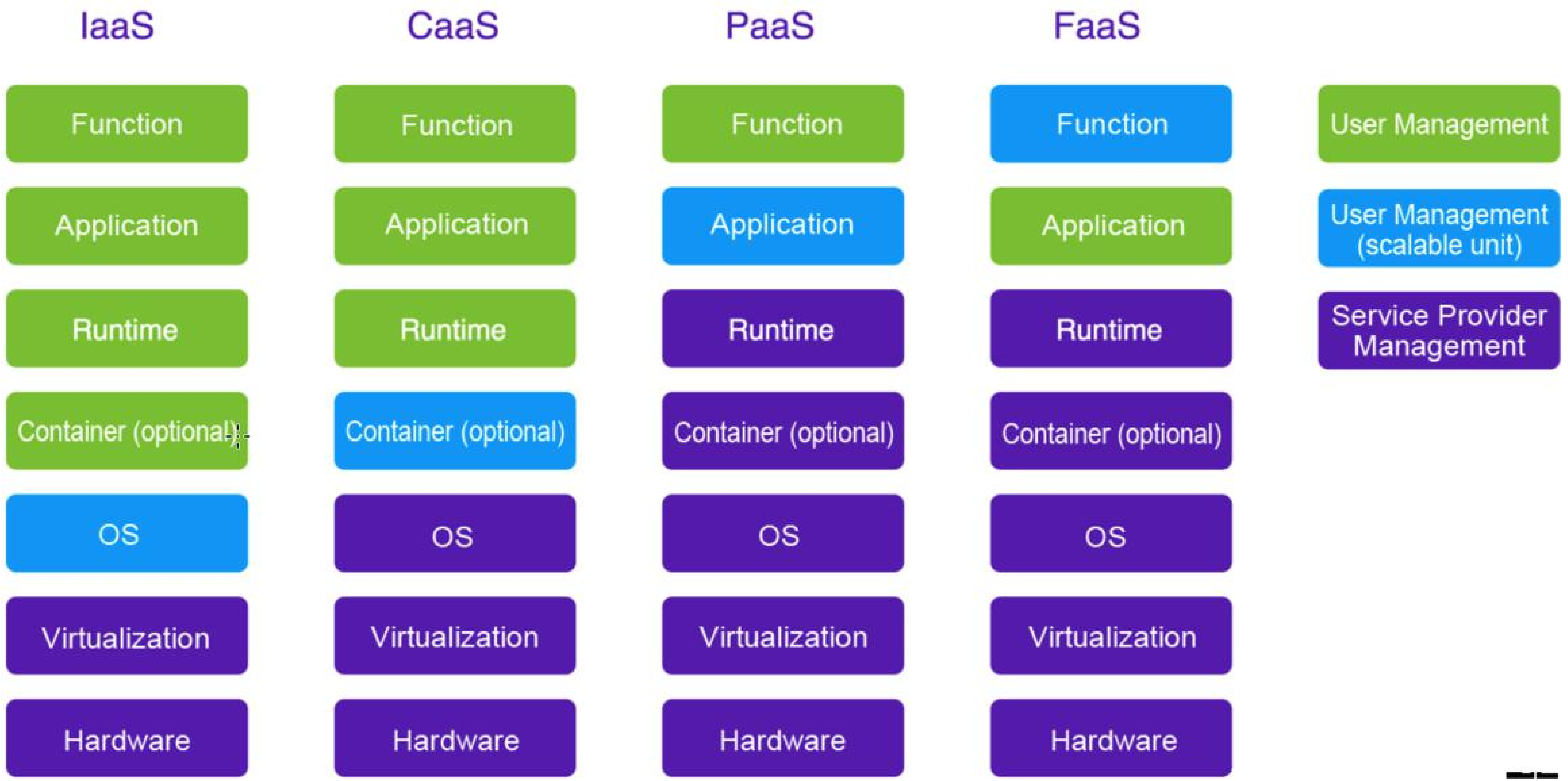
- Application/Services
- Infrastructure
- Architecture

All about
GRANULAR
and
SCALABILITY
and
GOVERNANCE

Application

- Serverless is lightweight event-based microservices.
- Google Cloud Functions is a **lightweight, event- based**, asynchronous compute solution that allows you to create small, single-purpose functions that respond to cloud events without the need to manage a server or a runtime environment.
- -Google Functions

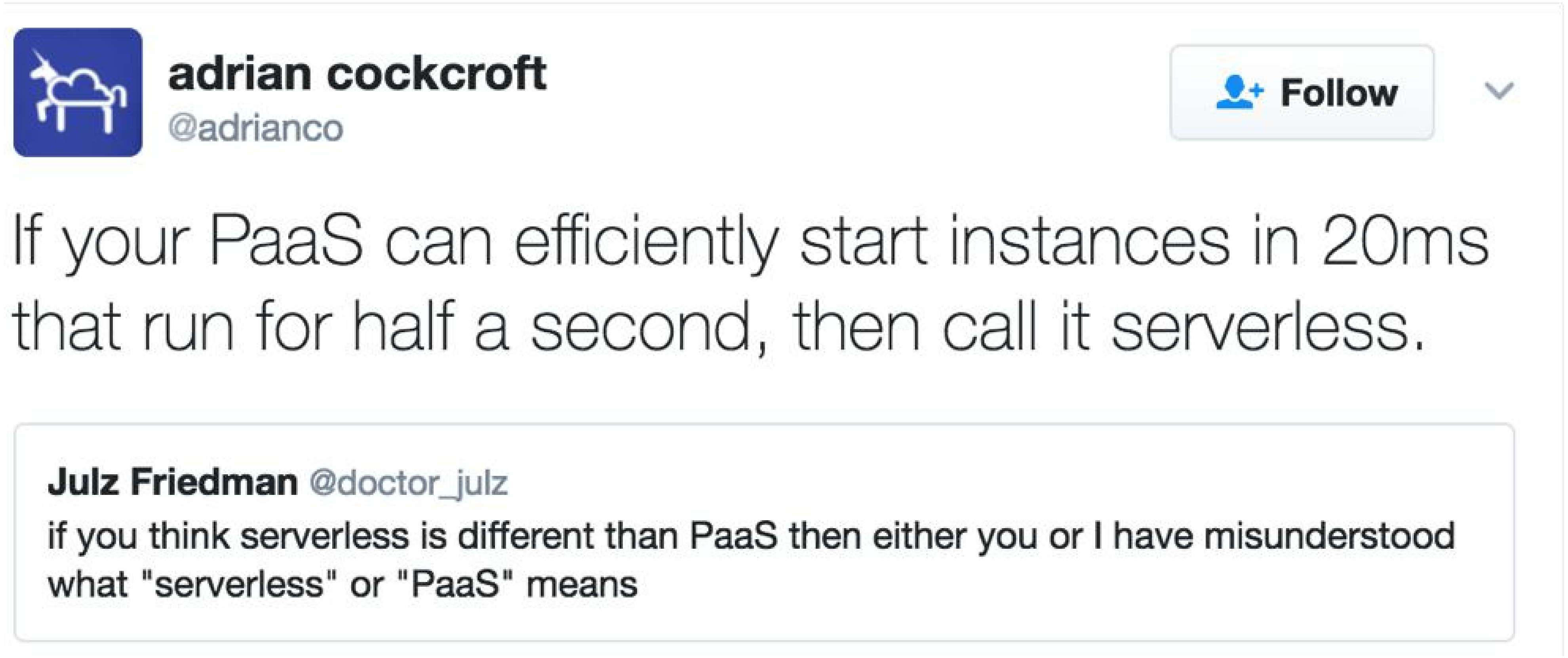
Infrastructure



Infrastructure

- Fully-managed by vendor
  - Without managing server system: OS, CPU, memory, network
  - Without managing server applications: apache, node.js, configurations
- Functions
  - AWS Lambda: Javascript, Python, JVM language, C#
- Deploy
  - BYOC: bring your own code. ZIP or code in console
- Scaling
  - Request based automatically
- Trigger by events
  - Defined by vendors: eg AWS S3, CloudWatch (schedule), Message Queue
  - Http/s requests: eg. API Gateway, Webtask

FaaS vs PaaS


adrian cockcroft
@adrianco

If your PaaS can efficiently start instances in 20ms that run for half a second, then call it serverless.

Julz Friedman @doctor_julz
if you think serverless is different than PaaS then either you or I have misunderstood what "serverless" or "PaaS" means

Architecture

- Stateless Function
- API gateway
- Event Driven Architecture (EDA)

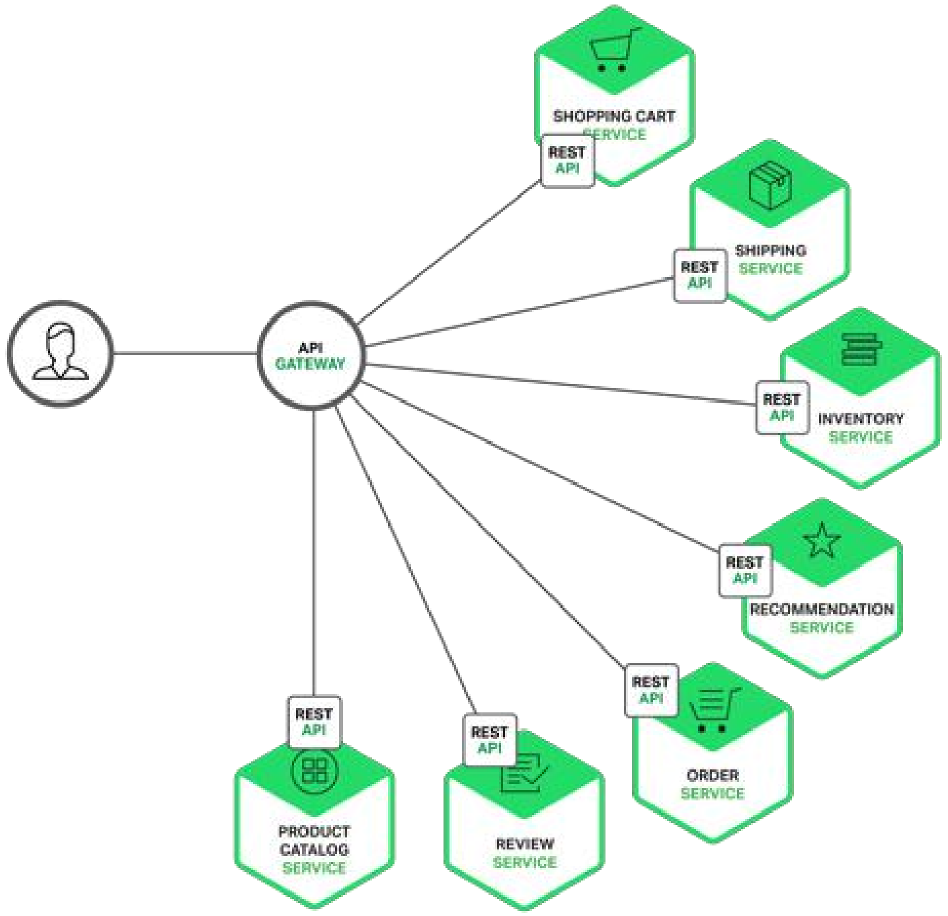
Pivotal

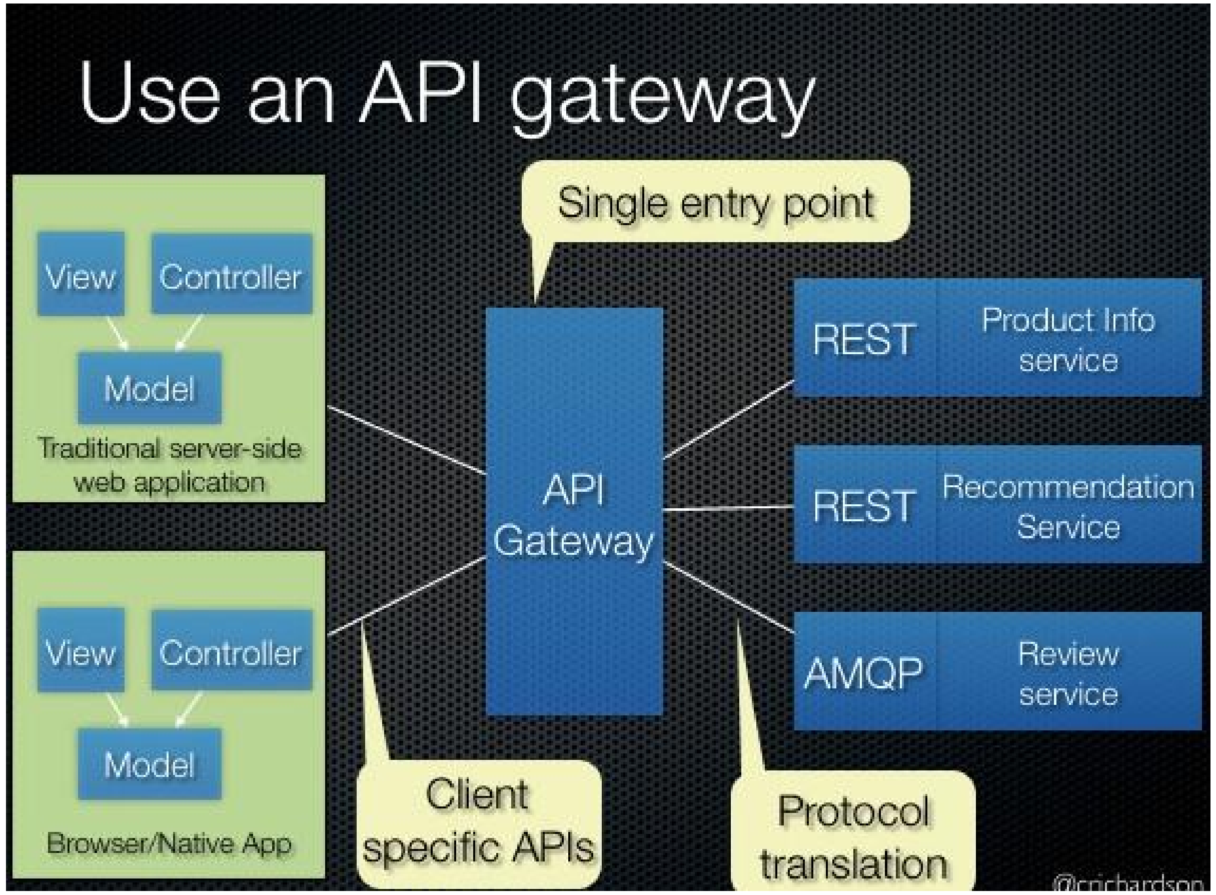
Stateless functions

- Processes are stateless and share-nothing
- Any data that needs to persist must be stored in a stateful backing service, typically a database.
- Don’t use “sticky sessions”
- https://12factor.net/processes

API gateway

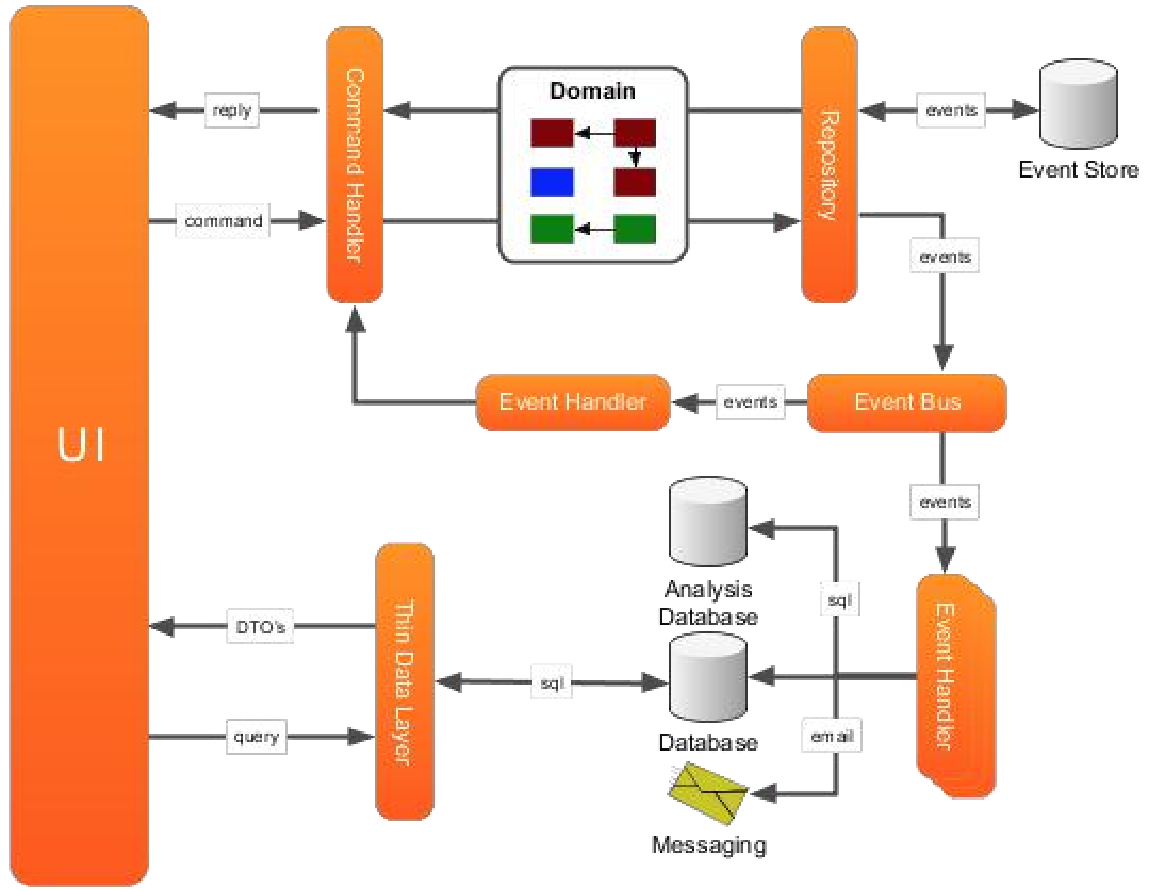
- Protocol/Auth/FlowControl/RequestControl…





EDA

- Workflow triggered and executed by event, not by hard coded function invocation.
- Typical code pattern
  - Pub/Sub
  - registry/deregistry
  - lightweight bus, dumb pipe/smart endpoint
  - Reactor/ Asynchronous



Pivotal

What’ s benefits

Benefits

- Reduce operational costs
  - Infrastructure cost: without pay as it is idle*
  - People cost: focus on function development
- Reduce development cost
  - BaaS: eg. Firebase, Auth0
  - Spend development time on business-specific code
  - Maximize iterations
  - Minimize dependences: IT Ops, DBAs
- Easier Operational management
  - Scaling benefits of FaaS
  - Reduced packaging complexity

What’ s drawbacks

Drawbacks

- Vendor Lock-in
- Vendor control
  - Startup latency: worst case 3 second for JVM
  - Execution duration: 300 second for AWS
  - DoS yourself: AWS 1,000 concurrent / second
- Versioning and deployment
  - If you have 20 functions for your application, how you handle deployment?
  - Any easy way to roll back all 20 functions atomically?
- Testing/Monitoring / Debugging
- Repetition library or codes

Pivotal

When using serverless

Where serverless make sense

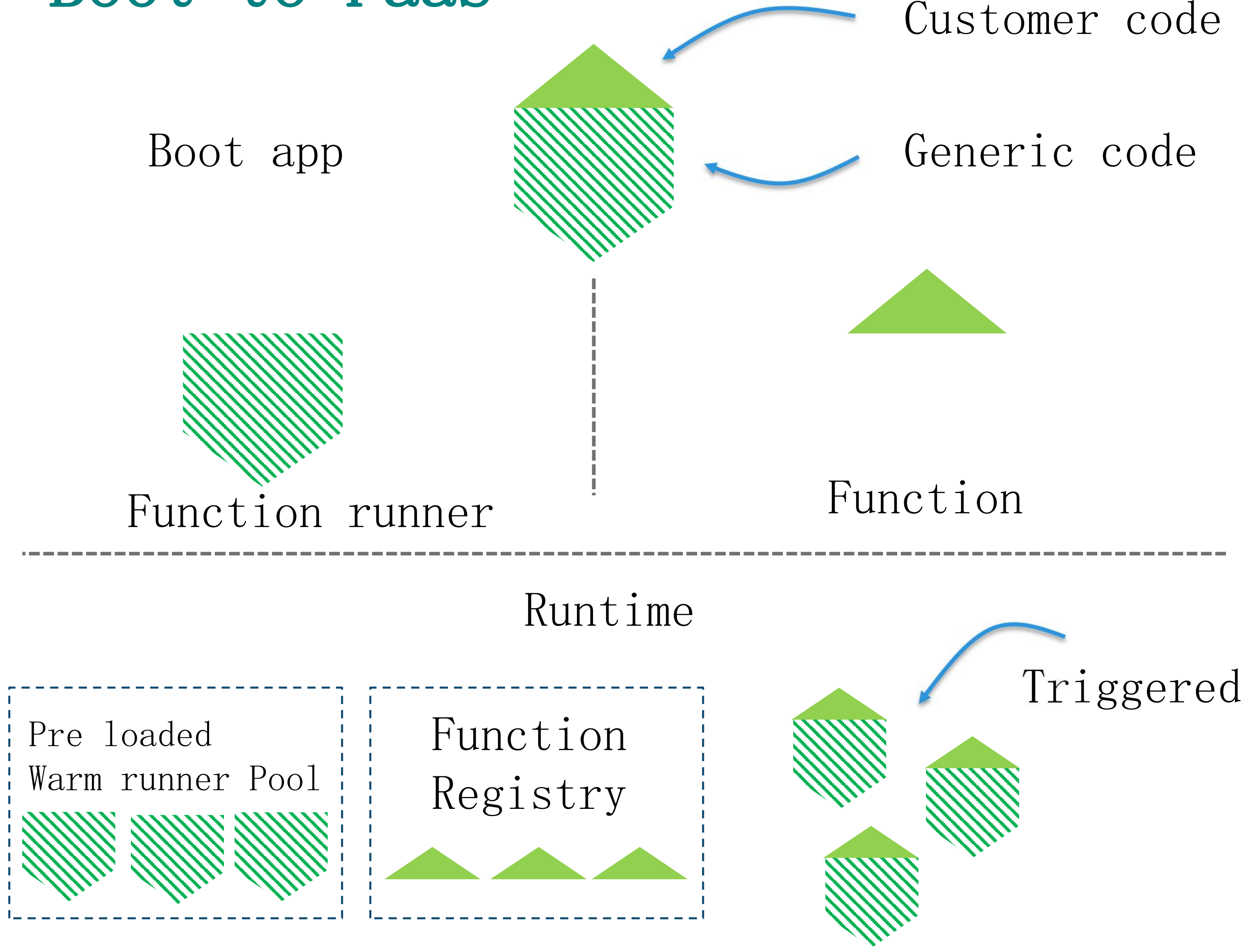
- **Fast** is more important than elegant.
- Change in the application's functionality and **usage is frequent.**
- Change occurs at **different rates** within the application, so functional isolation and simple integration are more important than module cohesiveness.
- Functionality is easily separated into **simple, isolatable** components.
- Each single-function has one **action.**

Pivotal’s approach

What is Pivotal offering

Spring Boot	Function runner and generic code pre loaded into “conainer”
Cloud Foundry Tasks	One-off scripts that run inside app container images
Spring Cloud Task	Library for short-lived Spring Boot apps
Spring Cloud Function	Upcoming framework for event-driven Java functions
Spring Cloud Data Flow	Orchestration of microservices and serverless tasks

Boot to FaaS



Boot app split into

1. Custom function jar
2. Function runner

FaaS Runtime maintains runners in warm pool

When triggered, runners load and then invoke

Cloud Foundry Tasks

What does it do?
Run an application or script within the context of an app container.

When should I use it?
Jobs for database migration, sending notifications, uploading data, running a batch job, optimizing a search index.

What else should I know about it?
Tasks "finish" versus regular apps that stay online. Ability to schedule tasks is coming. Team also looking to simplify invocation of app within the container.

Pivotal

Spring Cloud Task

What does it do?
Spring Boot annotation for short-lived JVM processes executed on demand.

When should I use it?
Makes sense to use as part of data pipeline (Spring Cloud Data Flow) or as part of event-driven actions connected to Spring Cloud Stream.

What else should I know about it?
Has a built-in audit framework that stores execution history. Integrates with Spring Batch, Spring Cloud Stream.

Spring Cloud Function

What does it do?
Wrap @Beans (or compiled strings) to consumers via HTTP or message broker.

When should I use it?
Build event-driven functions invoked by data changes, customer orders, webhooks, and more. Fits all the use cases applied to AWS Lambda or Azure Functions.

What else should I know about it?
Still under development! Will take packaged jars or individual strings and turn into available endpoints.

Spring Cloud Function + PCF

- The @Beans can be Function, Consumer or Supplier (all from java.util), and their parametric types can be String or POJO. A Function is exposed as an HTTP POST if spring-cloud-function-web is on the classpath, and as a Spring Cloud Stream Processor if spring-cloud-function-stream is on the classpath and a spring.cloud.function.stream.endpoint property is configured in the Spring environment. A Consumer is also exposed as an HTTP POST, or as a Stream Sink. A Supplier translates to an HTTP GET, or a Stream Source.
- Functions can be of Flux<String> or Flux<Pojo> and Spring Cloud Function takes care of converting the data to and from the desired types, as long as it comes in as plain text or (in the case of the POJO) JSON.
- Functions can be grouped together in a single application, or deployed one-per-jar. It’ s up to the developer to choose. An app with multiple functions can be deployed multiple times in different ″personalities″, exposing different functions over different physical transports.

Spring Cloud Data Flow

What does it do?
Compose a series of Spring Boot apps into a data pipeline.

When should I use it?
Replace batch processing with data stream processing. Invoke services or serverless tasks based on HTTP or messaging triggers.

What else should I know about it?
Included in upcoming release of Spring Cloud Services. Works on Cloud Foundry, K8s, Mesos, more. Has built-in connectors for Amazon S3, SFTP, GemFire, RabbitMQ, Cassandra, more.

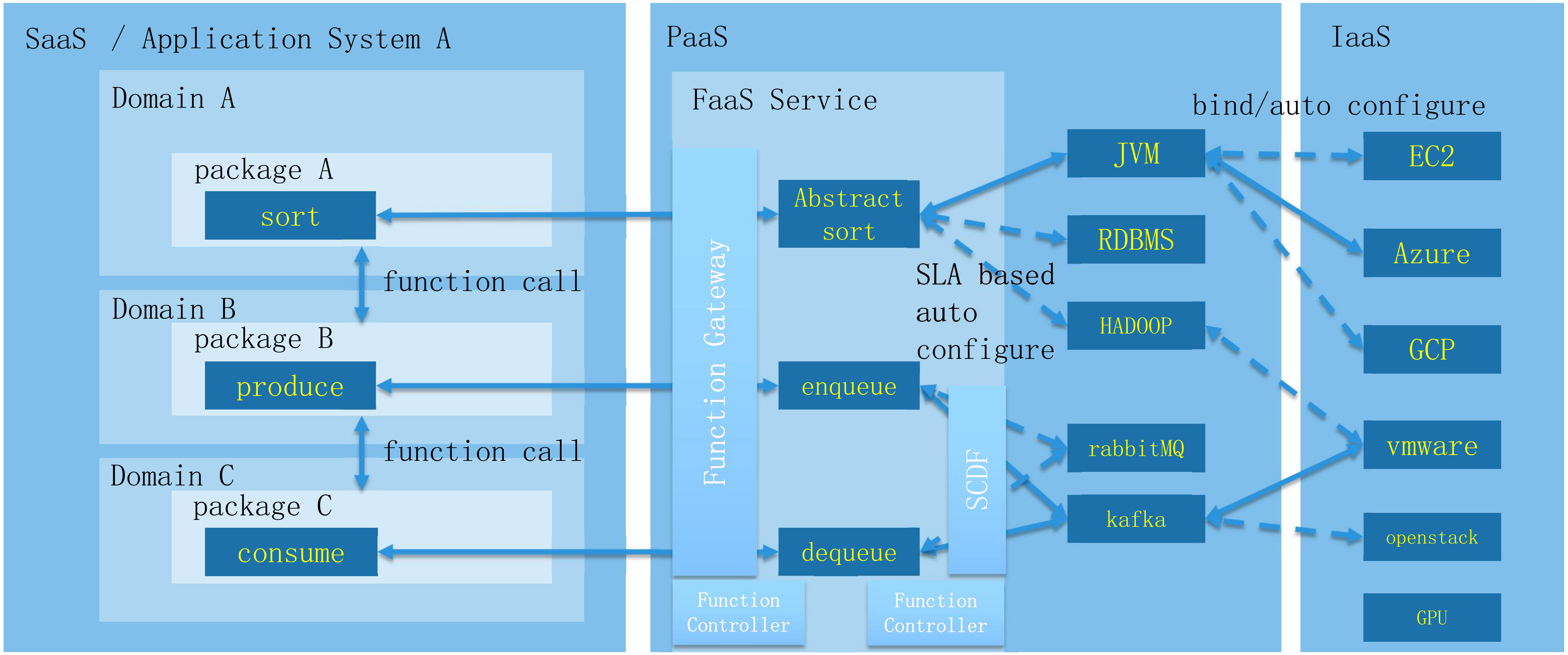
Pivotal Cloud Foundry for PaaS

- **Serverless doesn’t “kill” PaaS any more than containers do.** Serverless is an exciting addition to the options a developer has to build apps. It’s complementary. Also, Cloud Foundry plus buildpacks give customers a code-centric, “serverless” experience. It’s about solving the business challenge using whatever interaction model works best.

- **More types of apps are a fit for Cloud Foundry.** Cloud-native apps may be long-running web apps, IoT brokers that rely on TCP routing, batch jobs, or event-driven functions. Pivotal is focused on making Cloud Foundry the ideal place for *all* of them.

- **Portability matters.** The serverless trend is exciting, but keep an eye on where lock-in occurs. With Cloud Foundry and Spring, customers have choice of where they want to run.

All about abstract UP and break DOWN



Let’s build something

MEANINGFUL

Pivotal

Thanks!