微服务架构设计与Spring Cloud Function/Riff

王钧平（Rex）
Pivotal 云计算资深架构师

Agenda

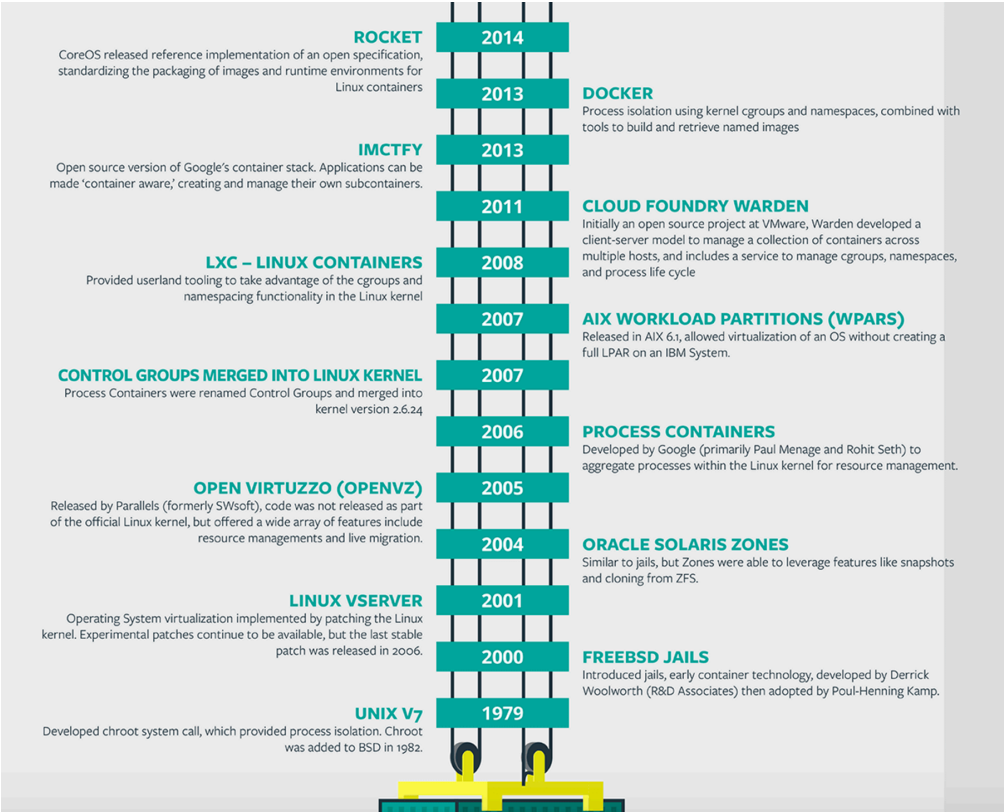- Introduction to Cloud Foundry Foundation

- From BOSH to CFCR

- What is riff ?

- Use Case

Moments in Containers History
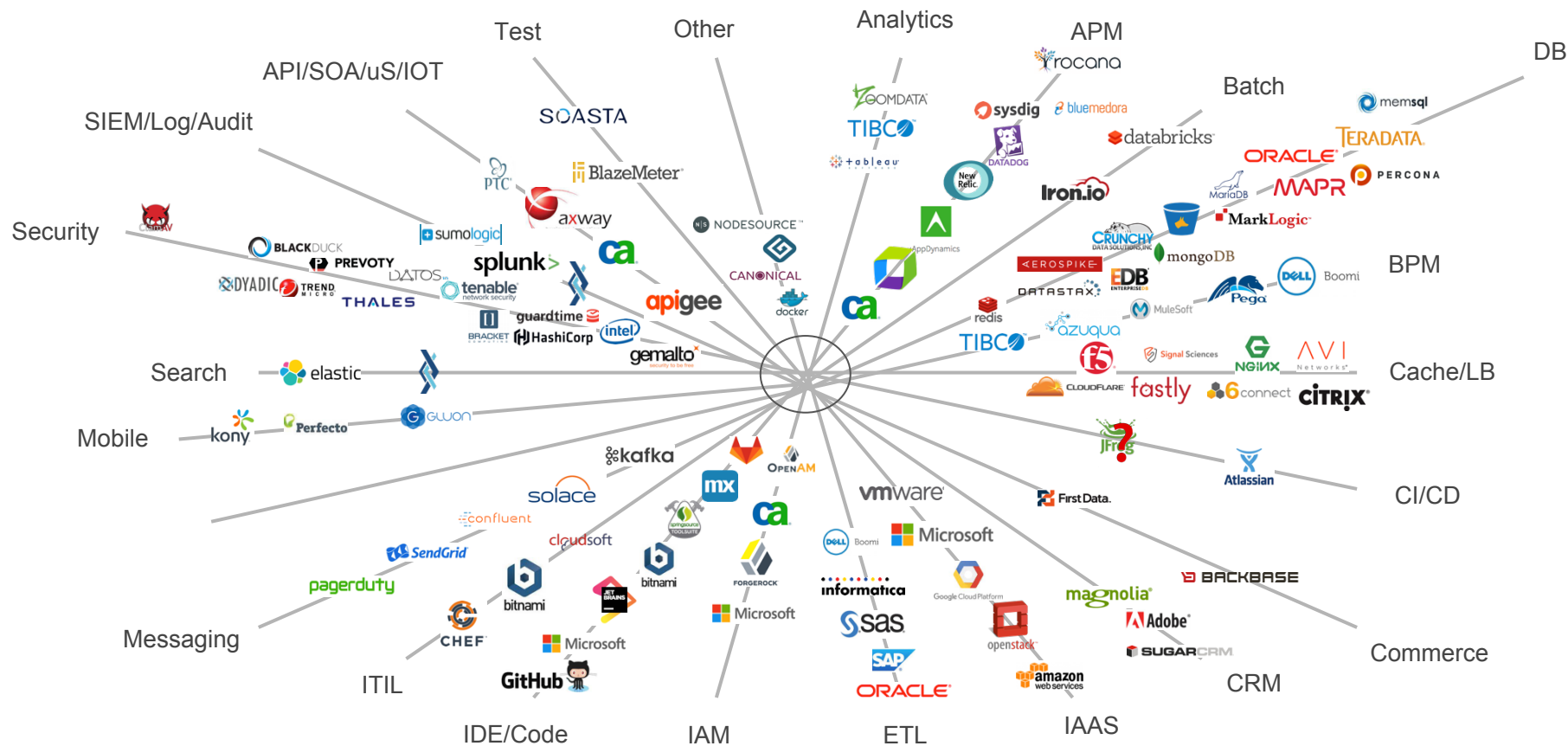

ROCKET — 2014
CoreOS released reference implementation of an open specification, standardizing the packaging of images and runtime environments for Linux containers

2013 — **DOCKER**
Process isolation using kernel cgroups and namespaces, combined with tools to build and retrieve named images

IMCTFY — 2013
Open source version of Google's container stack. Applications can be made 'container aware,' creating and manage their own subcontainers.

2011 — **CLOUD FOUNDRY WARDEN**
Initially an open source project at VMware, Warden developed a client-server model to manage a collection of containers across multiple hosts, and includes a service to manage cgroups, namespaces, and process life cycle

LXC – LINUX CONTAINERS — 2008
Provided userland tooling to take advantage of the cgroups and namespacing functionality in the Linux kernel

2007 — **AIX WORKLOAD PARTITIONS (WPARS)**
Released in AIX 6.1, allowed virtualization of an OS without creating a full LPAR on an IBM System.

CONTROL GROUPS MERGED INTO LINUX KERNEL — 2007
Process Containers were renamed Control Groups and merged into kernel version 2.6.24

2006 — **PROCESS CONTAINERS**
Developed by Google (primarily Paul Menage and Rohit Seth) to aggregate processes within the Linux kernel for resource management.

OPEN VIRTUZZO (OPENVZ) — 2005
Released by Parallels (formerly SWsoft), code was not released as part of the official Linux kernel, but offered a wide array of features include resource managements and live migration.

2004 — **ORACLE SOLARIS ZONES**
Similar to jails, but Zones were able to leverage features like snapshots and cloning from ZFS.

LINUX VSERVER — 2001
Operating System virtualization implemented by patching the Linux kernel. Experimental patches continue to be available, but the last stable patch was released in 2006.

2000 — **FREEBSD JAILS**
Introduced jails, early container technology, developed by Derrick Woolworth (R&D Associates) then adopted by Poul-Henning Kamp.

UNIX V7 — 1979
Developed chroot system call, which provided process isolation. Chroot was added to BSD in 1982.

Members of Cloud Foundry Foundation

BOSH Architecture

BOSH

Director

Agent Cmds

CLI

NATS

Registry

Postgres

HM

Blobstore

Cloud Provider Interface (CPI)

BOSH Managed Deployments

agent   Deployment "A" VMs

agent   Deployment "B" VMs
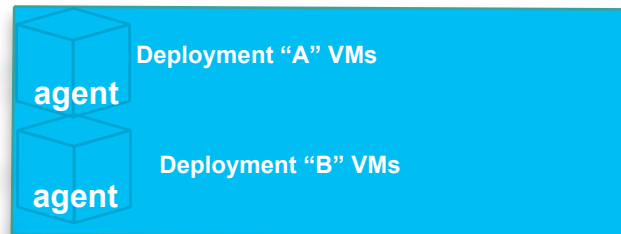
- Director uses CPI to create and configure VMs based on manifest
- Agent assists in configuration of VM on start up
- Director and Agent install and configure packaging on VMs
- Agent reports health and status of VMs to Director
- Director manages lifecycle of VM in case of upgrade or failure

High Availability

Fault-tolerance for masters, workers, and etcd nodes

Scaling

Auto-scaling of masters, workers, and etcd nodes

Health Checks & Healing

Routine health checks and self-healing of cluster

Lifecycle Management

LCM includes rolling upgrades to ensure workload uptime & application of CVEs

Their goal: pick the right runtime for each workload

CONTAINERS

Batches

EVENT-DRIVEN
FUNCTIONS

MICROSERVICES

DATA SERVICES

MONOLITHIC
APPLICATIONS

Container Orchestrator (CaaS)	Application Platform (PaaS)	Serverless Functions (FaaS)

IaaS
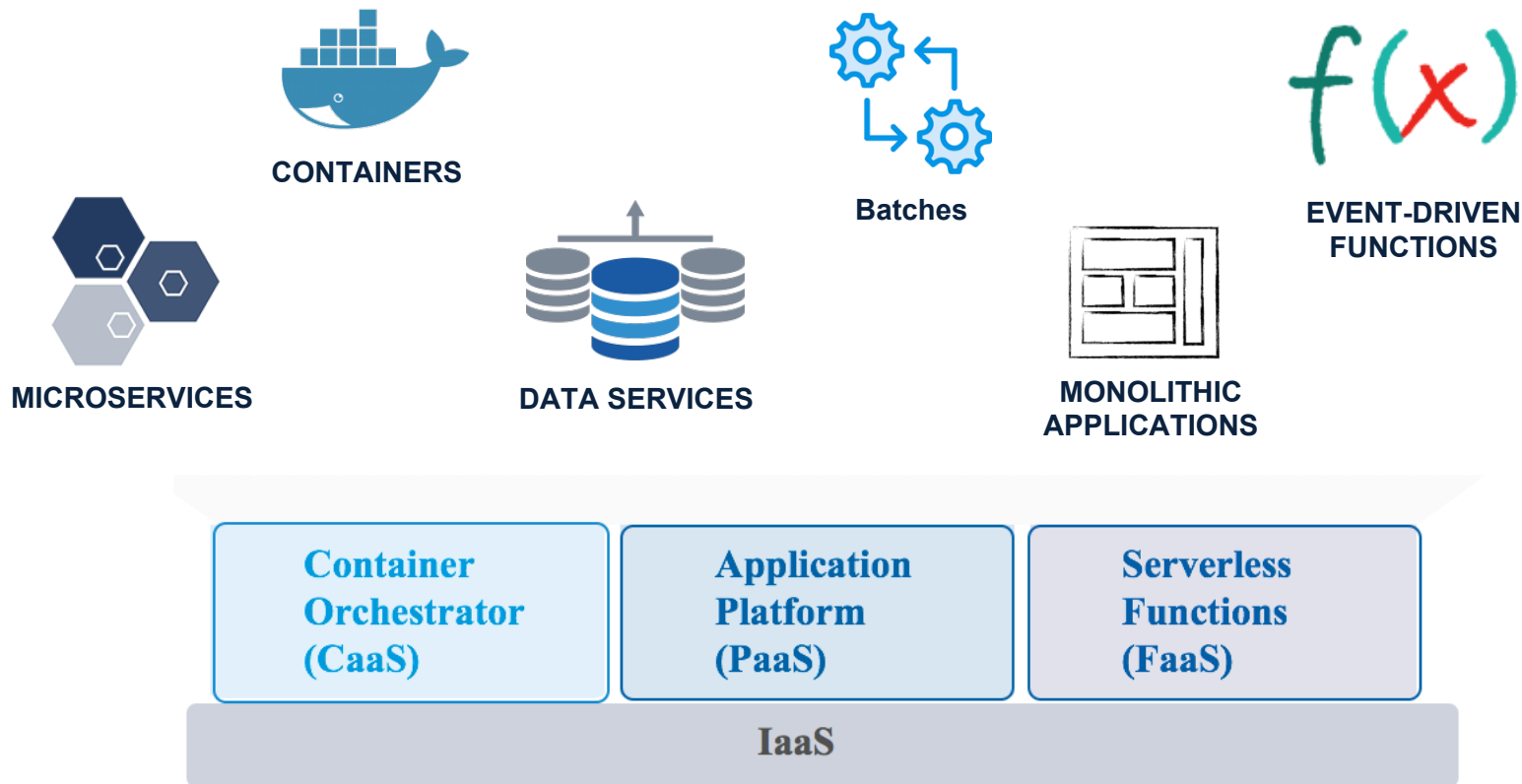
App Platforms + Functions

Comparing the experience for Devs + Ops

App Platform Abstraction	Functions Abstraction
runs **apps**	runs **functions**
push app to deploy server	register function and bind it to a trigger
server runs and waits for requests	function doesn't run until triggered
server listens to network	platform deploys and invokes functions
server handles lots of requests	functions handle **events** and then go away
scale out manually or by policy	**auto-scale** based on concurrent event load
pay per instance	pay per use – time & memory

FaaS Use Cases

Web Events

- Website back-end services like form post handlers, authentication, tracking and logging.

- APIs to back-end data services for mobile and web apps e.g GraphQL

- Webhook handlers

- Chat integrations

- Digital assistant services e.g. Alexa skills

Event-based Integration

- Scheduled tasks, ETL

- File processing e.g. images and videos

- Security scanning

- Complex Event Processing and Change Data Capture

- Monitoring, notifications and alerting

- Custom auth e.g. via API Gateway

Stream Processing

- IoT streams

- Log ingestion

- Event streams e.g. with Kinesis

- ML pipelines

riff is for functions

https://github.com/projectriff/riff

What is riff ?

riff provides developers with a service for executing Functions in response to Events.
Features
★ event streaming
★ polyglot
★ Kubernetes-native

Functions

f(x)

riff

Events

f(x)

f(x)

f(x)
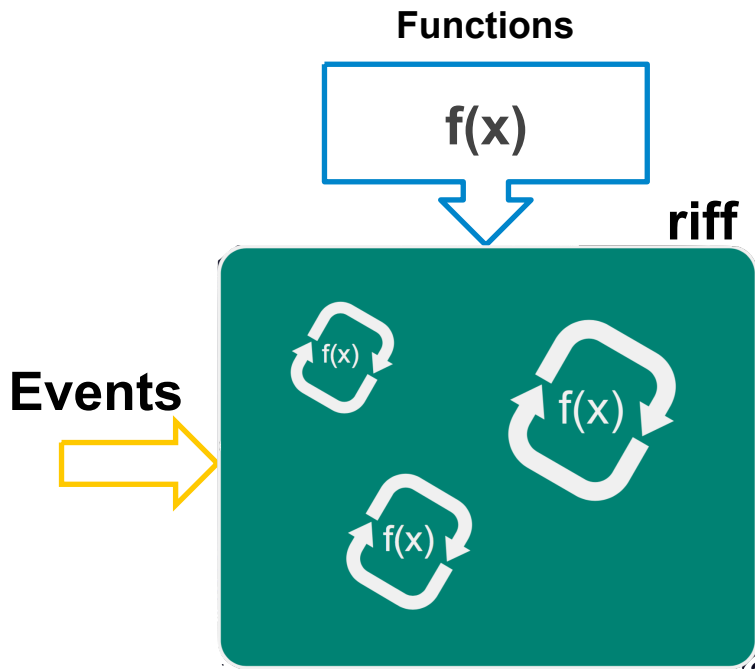
functions scale with events

riff function controller
★ interacts with K8s API
★ scales functions 0-1 and 1-N
★ monitors event-lag in Kafka

K8S API

Functions Controller
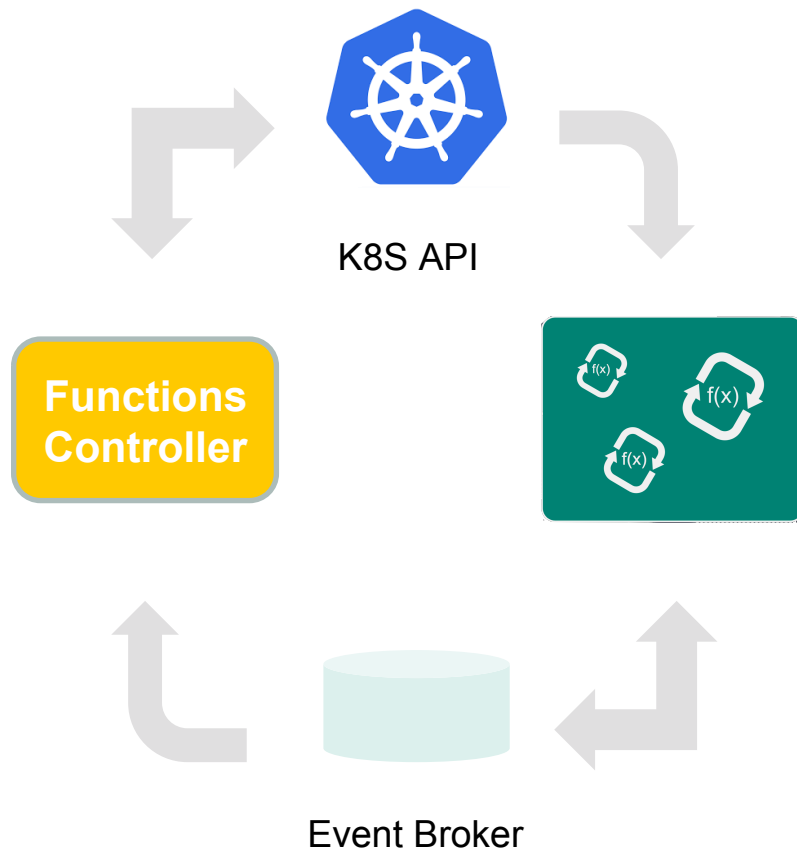
Event Broker

Sidecars connect Functions with event brokers

Function Pod

Sidecar Container

binder dispatcher

Function Container

invoker **function**

Broker-specific API

Event Broker

HTTP gRPC Stdio pipes
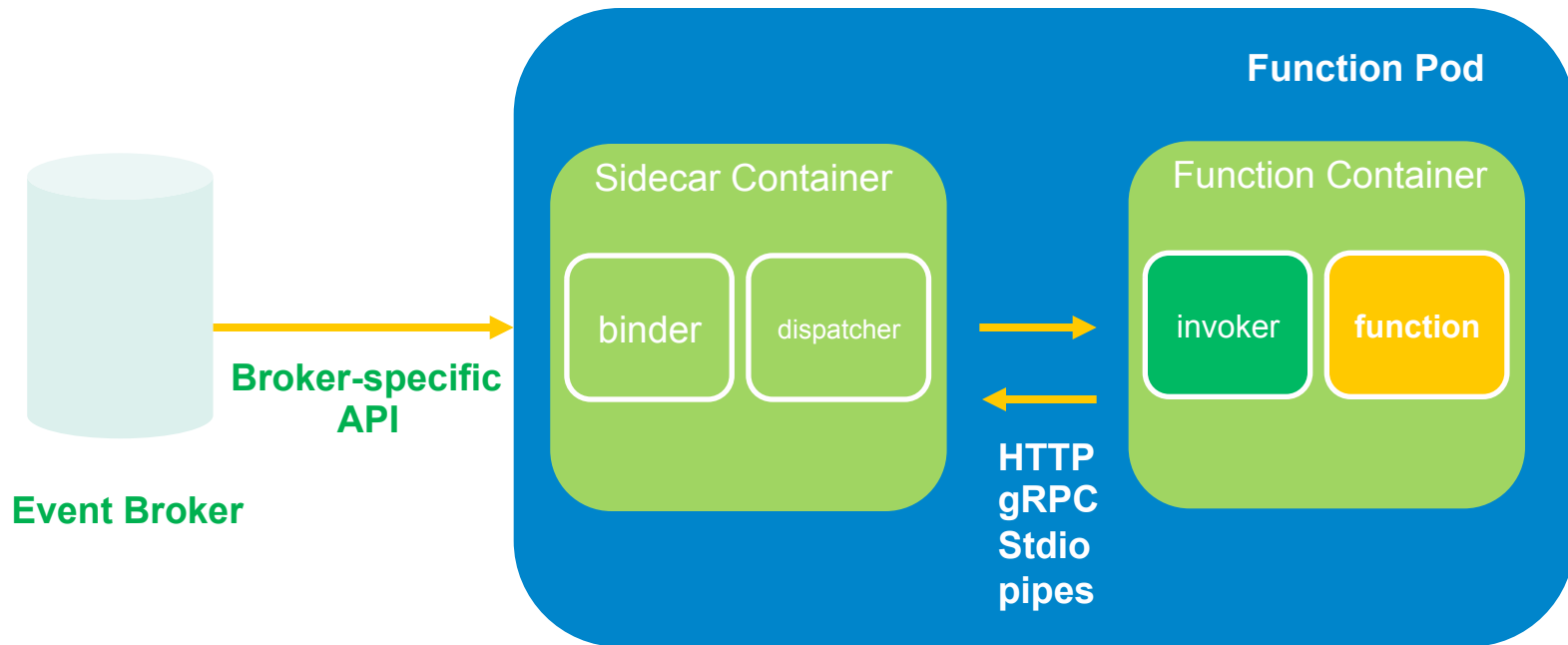
Installing riff via helm

Set up helm

```
kubectl -n kube-system create serviceaccount tiller
kubectl create clusterrolebinding tiller \
                            --clusterrole cluster-admin \
                            --serviceaccount=kube-system:tiller
helm init --service-account=tiller
helm repo add riffrepo https://riff-charts.storage.googleapis.com
helm repo update
```

Set up riff

```
helm install riffrepo/riff --name demo \
      --version 0.0.3-rbac \
      --set httpGateway.service.type=NodePort
```

Spring Cloud Function
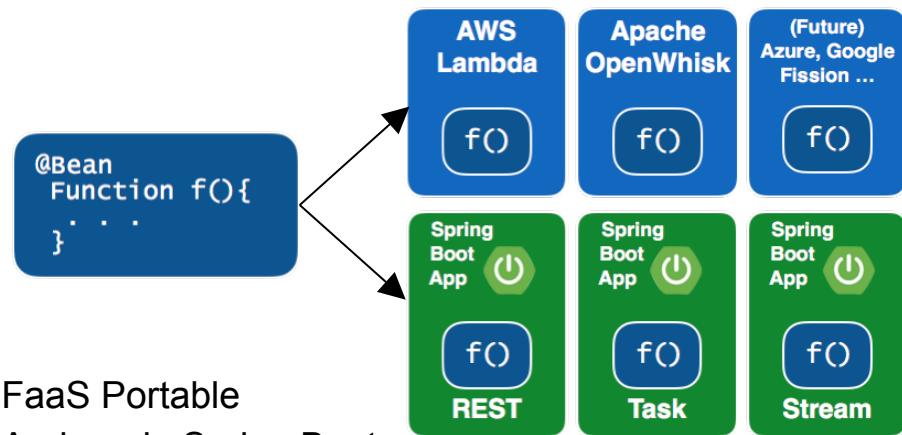
FaaS Portable
And run in Spring Boot
REST, Tasks, or Streams

• Focus on the implementation of business logic via functions

• Decouple the business logic from any specific runtime (HTTP, Message)

• Support a uniform programming model across FaaS providers

• Run standalone(locally or in a PaaS)

• Enable Spring Boot features on FaaS providers
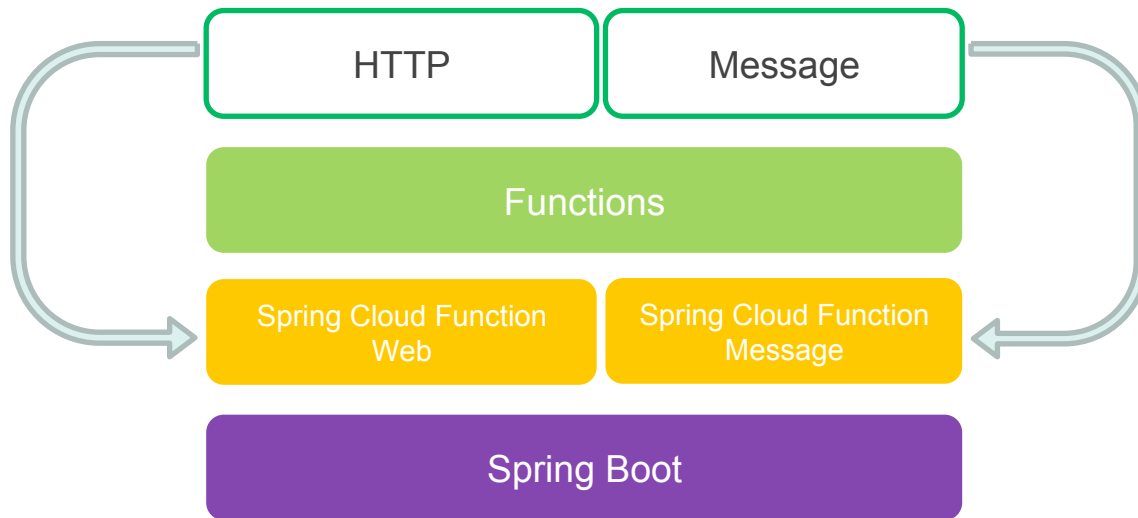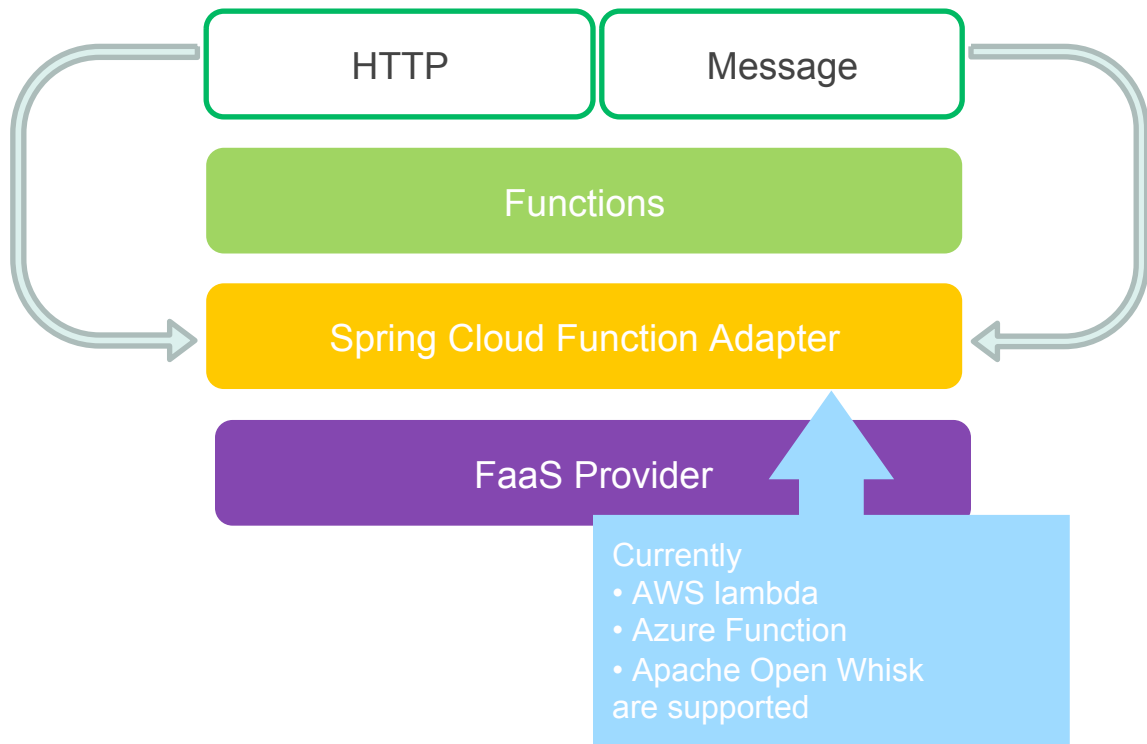
Source Code

```java
package functions;
import java.util.function.Function;
public class Greeter
        implements Function<String, String> {

    public String apply(String name) {
        return "Hello " + name;
    }

}
```

```java
@SpringBootApplication
public class App {
  @Bean
  public Greeter greeter() {
    return new Greeter();
  }
  public static void main(String[] args) {
    SpringApplication.run(App.class, args);
  }
}
```

Breaking in Monolith – Conway's law and Microservices
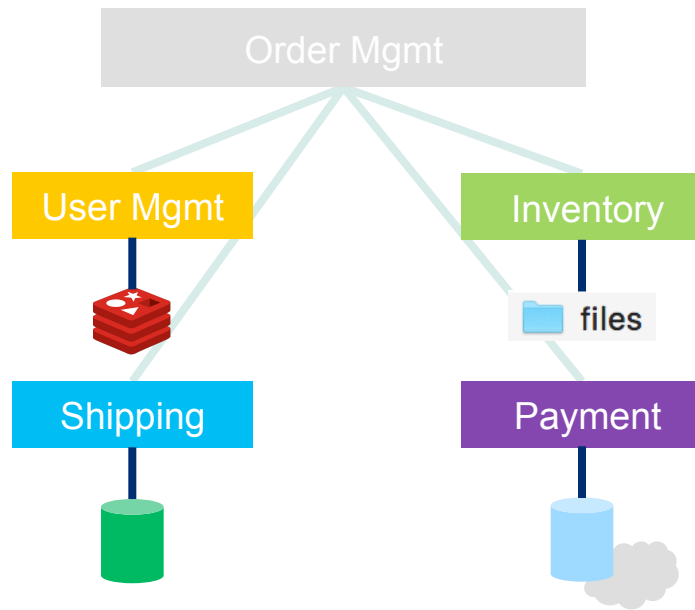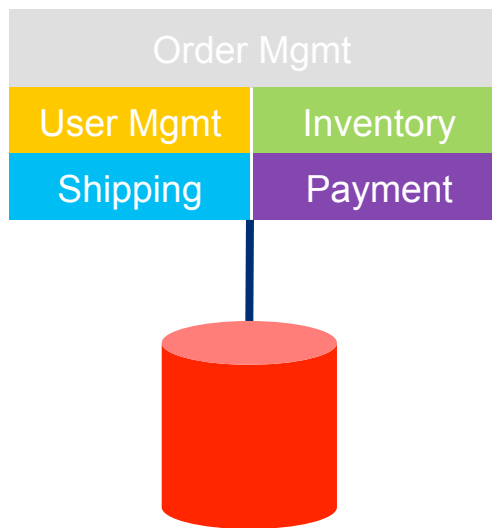
Microservices Architecture
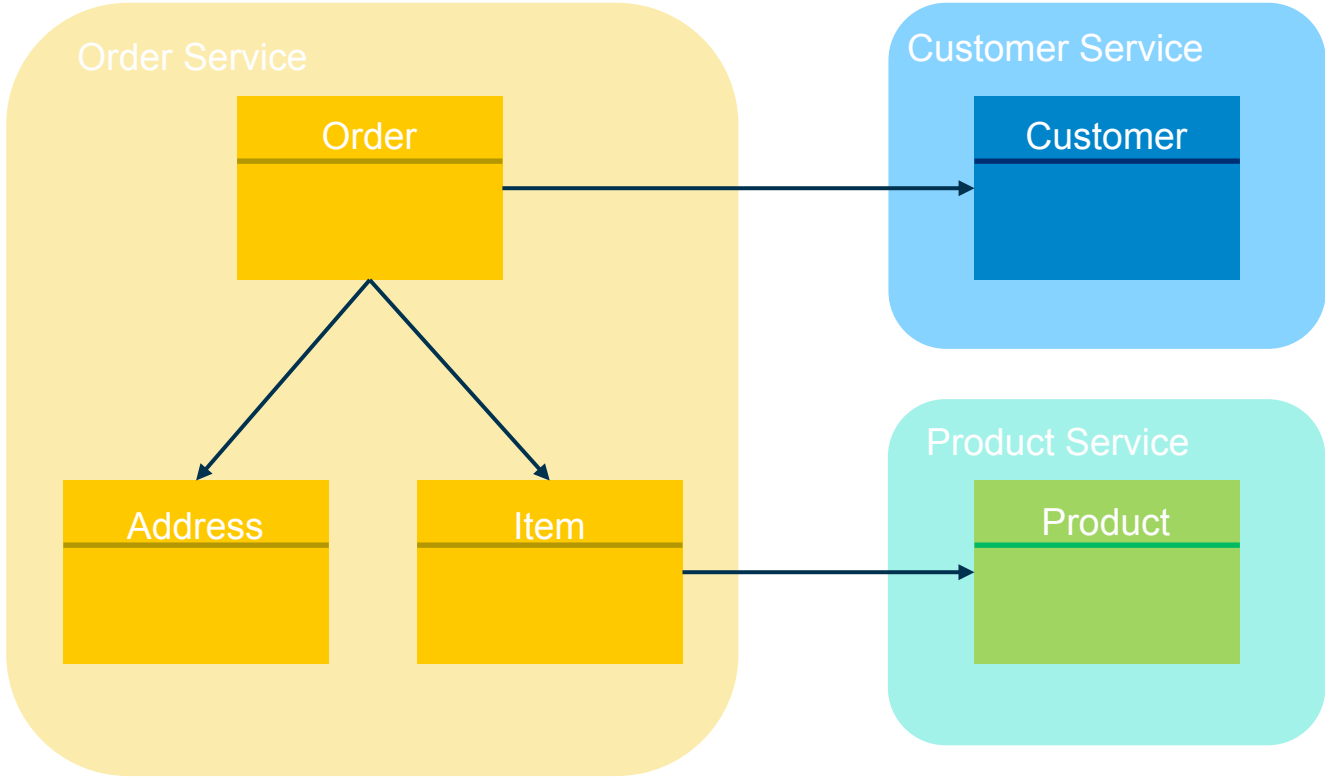
Transaction Flow – if-else
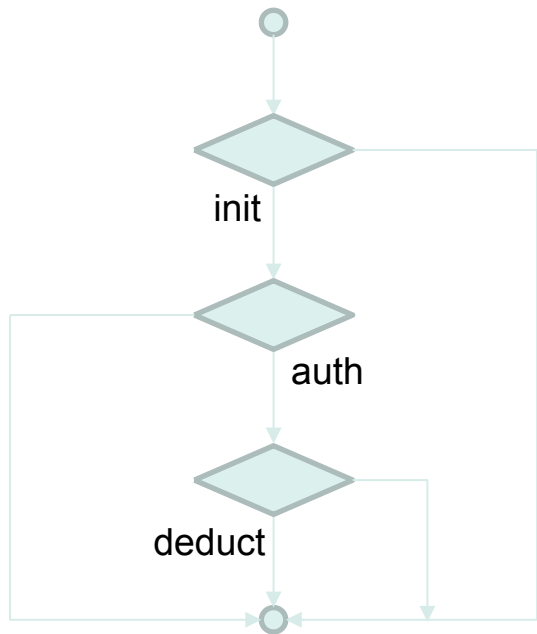

```
if(STATE_START.equal(CURRENT_STATE)){
    if()
        ...
    else{
        //檢核失敗
        return false;
    }
    ...
    //基本檢核成功
    CURRENT_STATE = STATE_INIT;
}

if(STATE_INIT.equal(CURRENT_STATE)){
    if()
        ...
    else{
        //檢核失敗
        return false;
    }
    ...
    //授權成功
    CURRENT_STATE = STATE_AUTH;
}

if(STATE_AUTH.equal(CURRENT_STATE)){
    if(){
        ...
    }
    else{
        //檢核失敗
        return false;
    }
    ...
    //交易成功
    return false;
}
```
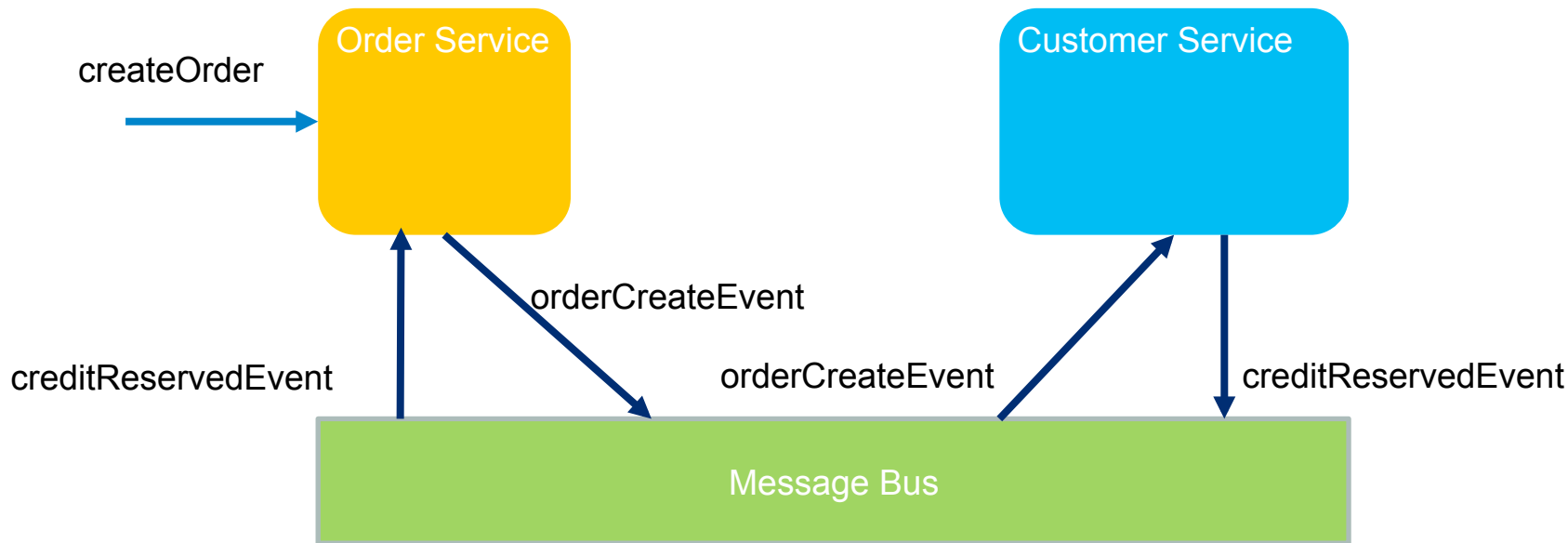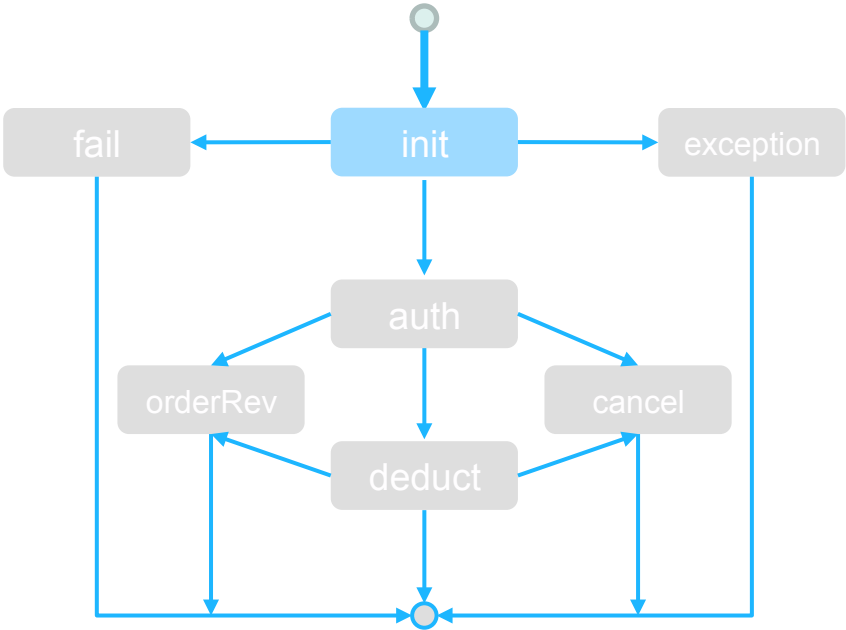
State and Command Pattern
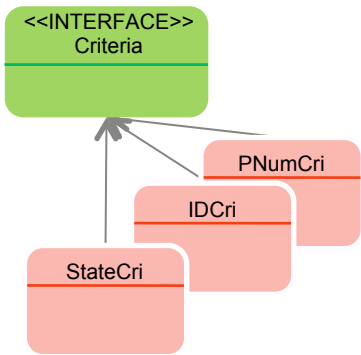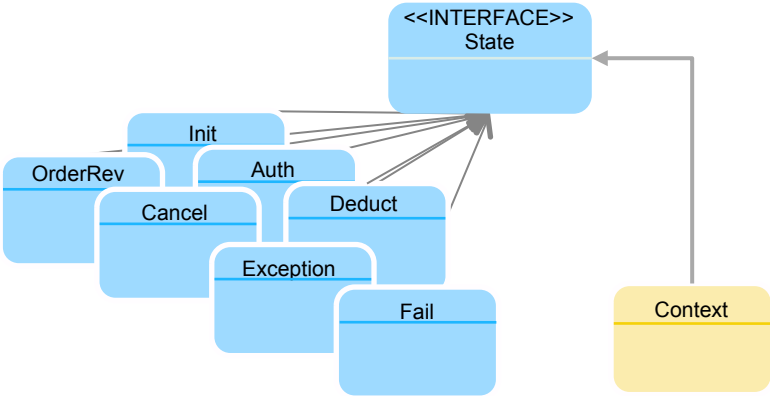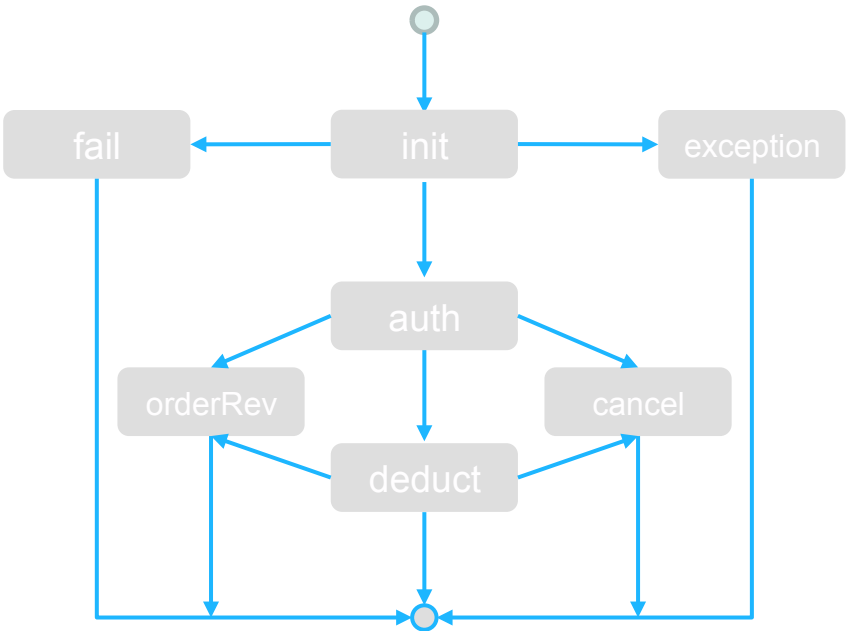
Domain Driven Design - Event Driven Architecture

Domain Driven Design – Event Sourcing

entity id	entity type	event id	event type	event data
1010	ORDER	901	INIT	180501180104
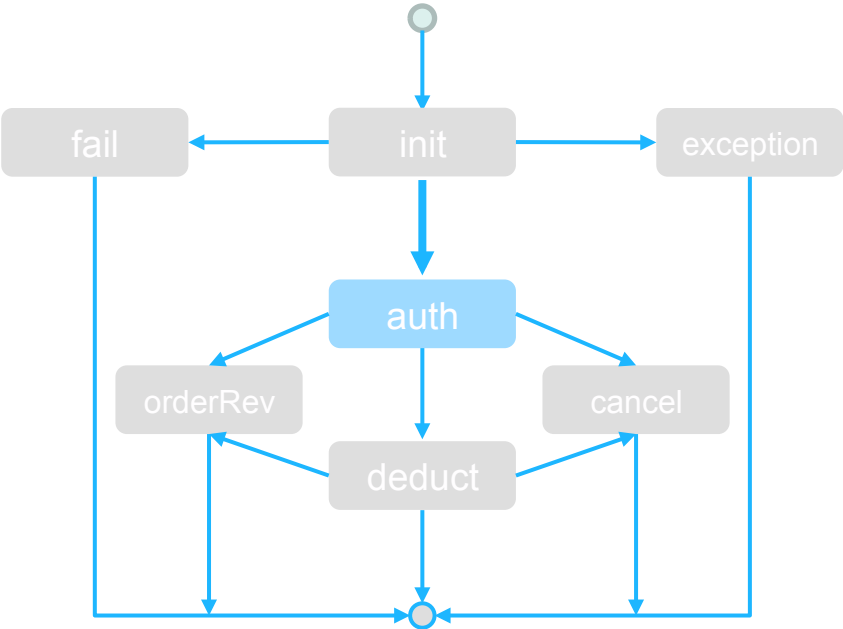
entity id	entity type	event id	event type	event data
1010	ORDER	901	INIT	180501180104
1010	ORDER	902	AUTH	180501180106
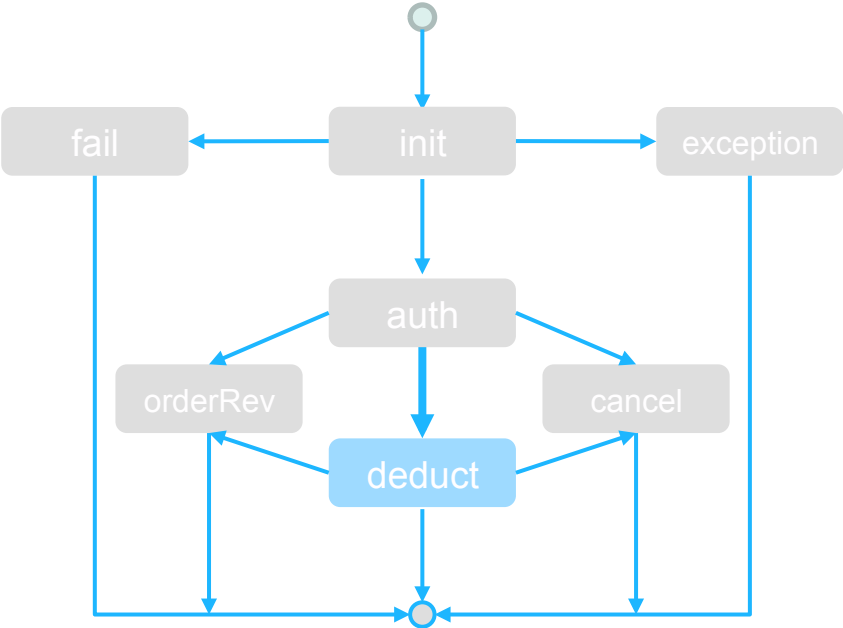
entity id	entity type	event id	event type	event data
1010	ORDER	901	INIT	180501180104
1010	ORDER	902	AUTH	180501180106
1010	ORDER	903	DEDUCT	180501180109

SpringOne Platform 2018

Pivotal.

Transforming How The World Builds Software