Service Governance in Production-ready containerized Cloud Foundry with Istio

Xuexiang Cui (Matt) cuixuex@cn.ibm.com
Gong Zhang (Grace) zhanggbj@cn.ibm.com
Xue Xiang Cui, cuixuex@cn.ibm.com

Xue Xiang Cui (Matt) is a Senior Software Engineer at IBM. He is the leader of IBM Cloud BOSH team responsible for the development of BOSH stemcell and CPI for IBM Cloud Infrastructure, and also plays a key role on the development of IBM Cloud Foundry Enterprise Environment (CF on Kubernetes) project.

Gong Zhang, zhanggbj@cn.ibm.com

Gong Zhang (Grace) is an Advisory Software Engineer with 6+ years work experience on Cloud Computing. She’s a contributor of BOSH Softlayer CPI, Kubernetes CPI and containerized Cloud Foundry solution. Currently, she is a developer working on IBM Cloud Foundry Enterprise Environment and also Istio integration.
Agenda

- Project Background
- Istio Introduction
- Motivations for CFEE and Istio Integration
- High-level Architecture and Implementation
- Future Vision
- Use Cases and Demo
Cloud Foundry is the industry-standard open source cloud application platform for developing and deploying enterprise cloud applications.

cf api https://api.<domain>
cf login –u admin –p password
cf push <app>
cf logs <app>
cf ssh <app>

App staging from cloudfoundry.org
**Cloud Foundry Enterprise Environment**: an isolated environment for hosting your Cloud Foundry apps with full admin control over configuration, capacity and access. Production-ready release 3.1.0.

### Core Features
- Runs on IBM Cloud container Service (IKS)
- Isolated Cloud Foundry environment
- Full administrative control
- Rapid provisioning and user onboarding
- Eirini (Technical Preview)
Istio

- lets you connect, secure, control, and observe services.
- Heavy investments from industry-leading companies such as Google, IBM, and Lyft.
- Production-ready Release 1.1.
- Active community.

Core Features
- Observability
- Fine-grained traffic management
- Security
- Platform support
- Integration and customization
Motivations for CFEE and Istio Integration

- Observability:
  - Service mesh status (CFEE components converted to microservices)
  - Metrics, logging and tracing
  - Fine-grained traffic management

- Operate & Manage:
  - Canary deployment
  - Blue/green deployment
  - Circuit breaking, fault injection, timeout & retry etc.

- User

- Public Env
  - Network & infra
  - Core components
  - Services
  - Applications

- Dedicated Env

- Local Env
  - Environment hosted on VM
  - Environment deployed by IBM
  - Monitored and operated by IBM

- Monitoring
  - Metrics
  - Alerts

- Operator

- IBM

- Istio

- CFEE
  - Cloud Foundry Enterprise Environment

- User/Operator/Admin

- Environment hosted on Armada
- Self-maintained tools for customer to create & scale their own CFEE
- Embedded monitoring, and operated by customer themselves
Motivations for CFEE and Istio Integration

**Operation**
- hard to trouble shooting
- limited observation

**Traffic Management**
- Limited control over traffic

**Metrics, logging & tracing**
- No service mesh layer data collected

**External Services**
- Barely no governance

**Certificate Management**
- Self maintained cert management

---

Istio

- Fine-grained traffic management
- Introduce telemetry for service traffic
- Bring external services into mesh and manage
- Manage and rotate certificates
Difficulties to bring Istio Into CFEE

• Containerized traditional CF
• Istio conventions required
• Complicated CF traffic
• Based on IKS platform
• Code change required to fully-enable tracing

How to resolve them

• Move to native micro-service architecture
• Refactor and comply
• Refine CF traffic
• Selectively integrate with IKS
• Enable tracing by component
Future Vision

• Contribute to open source containerized CF community leading by SUSE & IBM (in progress)
• Bring Istio into Eirini who Integrate Application Runtime and Kubernetes
• Leverage Istio mesh expansion to bring operational experience back to VM-based CF
Use Case – Get Operational Insights: as an operator/administrator, I’d like to:

Get insights of the overview of CFEE status and observe how CF components collaborate on a request with complex behaviors.

With visualization of the service mesh topology, observe Istio behaviors like circuit breakers or request rates and traffic shifting on CFEE etc.
Use Case – Inspect by enhanced Monitoring & Logging & Tracing: as an operator/administrator, I’d like to:

Gather traffic metrics and collect customized logging for all CFEE components, not only underneath Cloud Foundry metrics and component and application logging.

With tracing abilities, tracking down how user request flow though each of our components, and figure out where the request failed and find the performance bottle neck.
```go
func (h *DesiredLRPHandler) RequestLRP(logger lager.Logger, w http.ResponseWriter, req *http.Request) {
    logger = logger.Session("request-lrp")
    span := opentracing.GlobalTracer().StartSpan("RequestLRPAuctions",
    defer span.Finish()
    ctx = opentracing.ContextWithSpan(ctx, span)

    incoming_headers = [ 'x-request-id',
        'x-b3-traceid',
        'x-b3-spanid',
        'x-b3-parentspanid',
        'x-b3-sampled',
        'x-b3-flags',
        'x-ot-span-context' ]
```
Use Case – Fail Gracefully by Circuit Breaking: as an operator/administrator, I’d like to:

Retrying against an already-slow service just makes the service and whole environment to lag. With circuit breaker, eliminate slow CFEE components until they recover and avoid bringing down the whole CFEE environment, without changing CFEE component source code or configuration.
Use Case – Blue/Green Deployment: as an administrator, I’d like to:

apply Blue/Green Deployment for CFEE, instead of simple K8s rolling out, so we can control and distribute the Blue and Green version based on various properties of the request to do verification, and flip over the traffic to complete upgrade.

Use Case – Canary Deployment: as an administrator, I’d like to:

apply Canary Deployment for CFEE, instead of simple K8s rolling out, so we can have the ability to do fine tuned traffic shifting, to verify on actual production traffic, to release a new version to subset of users and finally observer and complete the whole upgrade.
Backup
Backup

Monitoring System

Istio Mixer
Prometheus
Vizceral
Istio Citadel
Istio Pilot

api
bbs
xxx
xxx
xxx
xxx

operate/manage
Use
provision
deploy
Admin Console

CFEE User/Operator/Admin
Thank You!😊