Clover: cloud native computing for NFV

Wenjing Chu

Stephen Wong

10/25/2017

What is Clover?

- To support VNFs developed as cloud native applications.
- “Cloud Native” has various definitions, e.g.
  - The 12 factor app: https://12factor.net/
  - CNCF definition: https://www.cncf.io/about/charter/
    - Container packaged (e.g. Docker)
    - Dynamically managed (e.g. Kubernetes)
    - Micro-services oriented (our focus)
  - Continuous delivery (our focus)
- In OPNFV, projects like container4nfv are covering both container (e.g. docker) and orchestration (e.g. k8s or openstack+k8s or other variants) infrastructure. These are out of scope, and it is a dependency for Clover.
- In Clover, we focus on:
  - Micro-service: employs loosely coupled microservices which are deployed within lightweight container environments that employ low-overhead orchestration, runtime and networking services
  - Continuous delivery (CD): Written and released using lean DevOps methodologies. Operates on continuous release cycles, in that new features can be independently developed, tested and then released/deployed.

Problems to solve (1)

- Monolithic infrastructure slows development and constrains scalability and reliability, so the industry has been trending on micro-services based architecture [micro-services].
- Growth of micro-services, however, often leads to 'container sprawl': service-to-service communication becomes unwieldy and hard to manage; integration and testing permutations quickly multiply; uniform policy enforcement becomes a lot harder…
- These issues can be esp. prominent in service provider and NFV use cases.
- A dedicated service layer (Service Mesh) agnostic to individual app and deployment and built on top of k8s's existing pod and service constructs and orchestrator functions, is designed to overcome these challenges.

Container sprawl

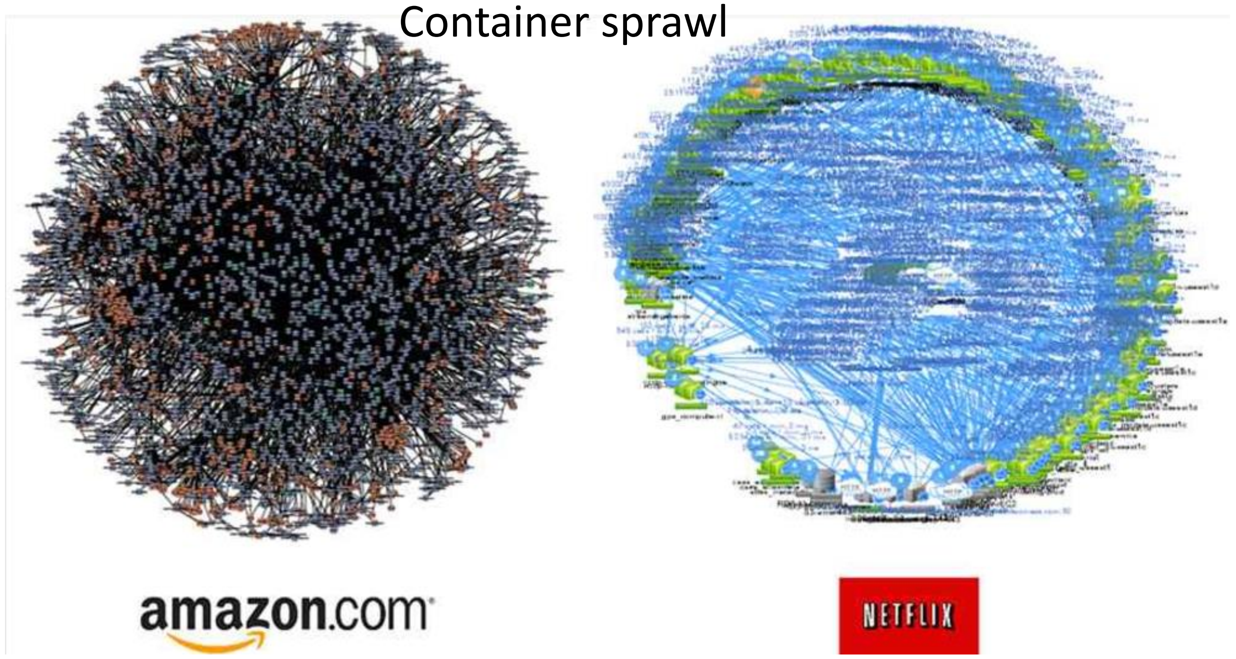
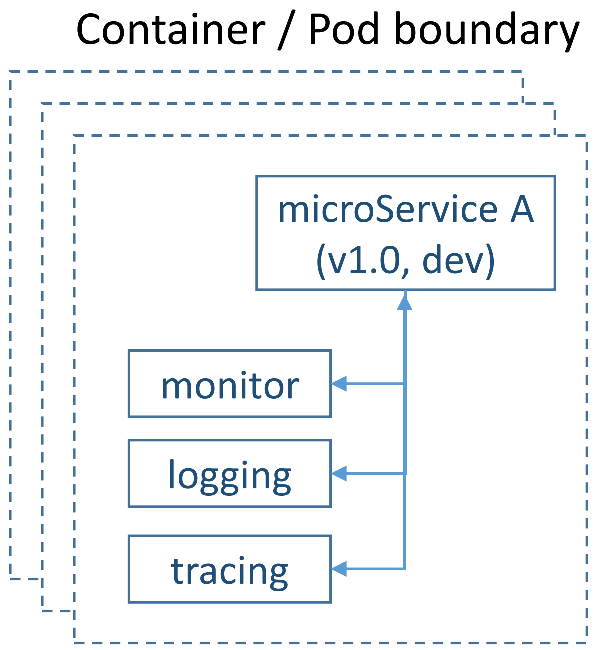


Image credit: Appcentrica.

- A service mesh to handle/manage internal service-to-service communications
- Dynamic request routing on live deployment e.g. canary, A/B testing, rollout etc.
- Fault tolerance, e.g. circuit breaker
- Richly featured service discovery and load balancing
- Global control and policy mechanisms that are made visible to management higher layer (e.g. mano)
- Globally distributed diverse environments

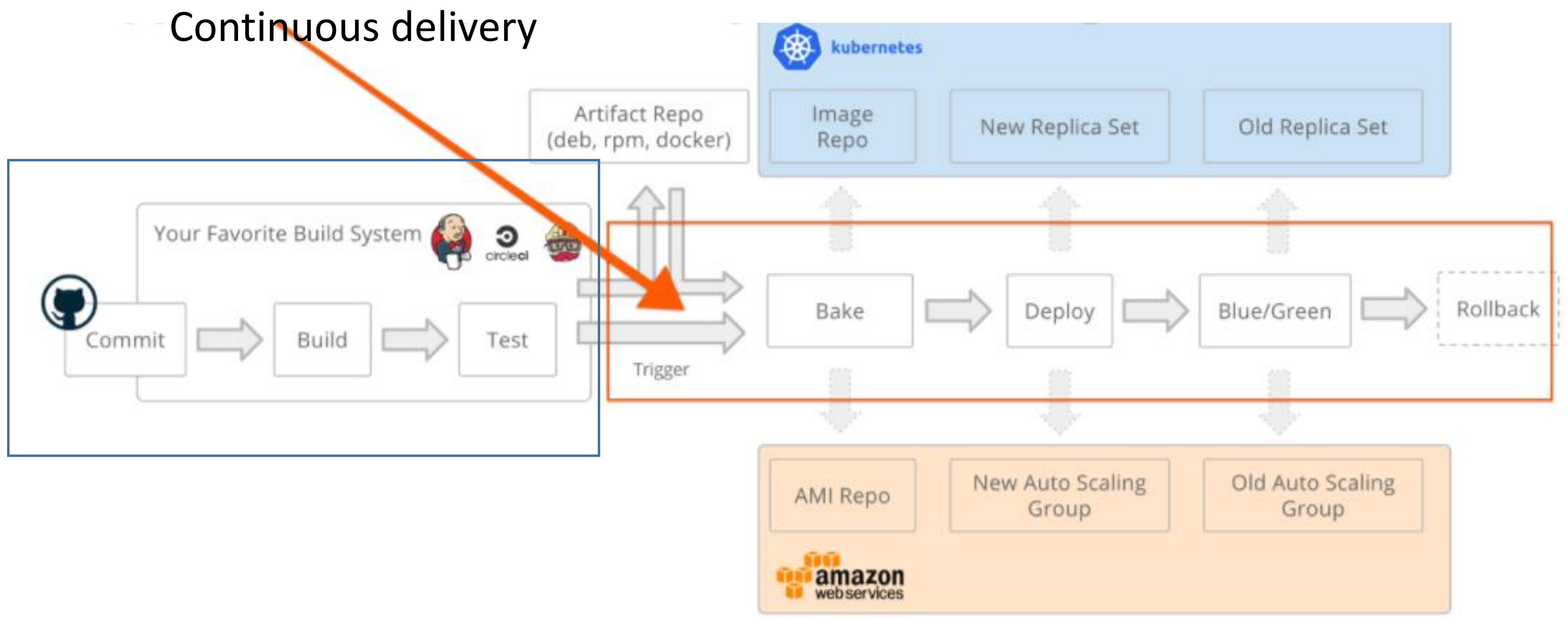
Problems to solve (2)

- The service providers found operating a large set of micro-services challenging without a common set of tools that give them visibility and traceability. This challenge is esp. relevant when applications are developed independently (by different teams, vendors or communities), or e.g. legacy software components are containerized and brought in to the mix. A common set of reusable services, such as monitoring, logging, tracing…, agnostic to development and deployment environments is needed.
- NFV specific use cases demand additional common utility services that are outside of the general computing applications. Such reusable services may include network tracing, nat, dpi etc. Storage related common services for NFV are also an area to investigate.



Problems to solve (3)

- Continuous delivery – a core benefit of cloud native applications is that its components (micro-services) can be updated live independently. Currently CI/CD pipeline in OPNFV largely stops at CI level.
- Building safe, low risk, platform independent and at scale continuous deployment is an important challenge facing NFV use cases. We need to address it in collaboration with xci/releng and upstream e.g. cncf cross-cloud ci.



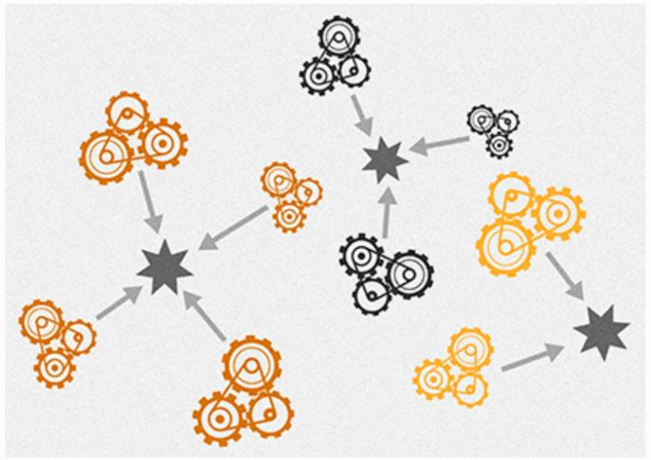
Problems to solve (4)

- NFV specific use cases introduce new challenges to solve that are different from the general cloud native cases. We need to investigate and identify the gaps and come up with solutions.
  - E.g. network needs for gateway (intermediate systems) common in NFV use cases. Potential network performance issues, security issues. Investigate storage related issues.
  - Investigate edge related challenges, in collaboration with edge related projects.
  - Investigate enhancements or additional common services for NFV
  - Interfacing with other OPNFV or LFNF projects (e.g. ONAP)
  - Best practice of developing cloud native VNFs
- Contribute upstream for the identified NFV specific features.

Scope and Dependencies

- Pharos pod:
- 1 identified
- 1 more for long duration test desirable later

Dependencies:
Pharos
Lab-aaS
Releng/xci
Container4nfv
Installers
K8s-* scenarios
Test projects
Doc …



Cloud native NFV in micro-services architecture

Container environment (e.g. docker, others, unikernels…)

Infrastructure (bare metal, VM, edge devices etc.)

Dynamic management/VIM
(Kubernetes)
(K8s+Openstack)
(Others)

Main focus area:
- Service mesh
- Common services
- Continuous delivery
- NFV gaps and features

Out of scope.
Rely on existing projects and upstream

Clover framework

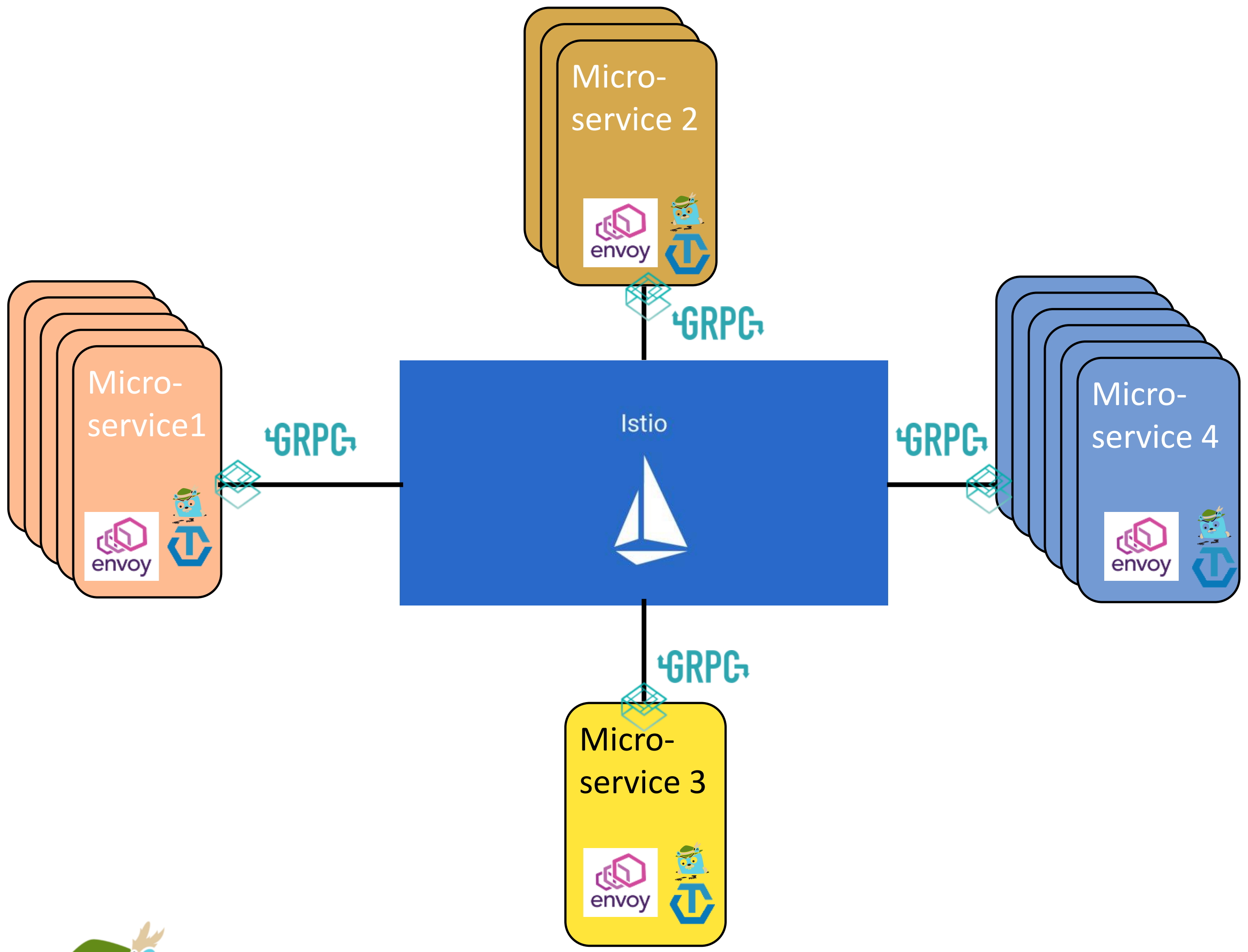
Out of scope; existing projects

Container

Orchestration

Continuous delivery, e.g.

Reusable Services, e.g.



Micro-service service mesh, e.g.

Some of the target computing environments

Tiny

OPNFV

PHAROS

Small
(Edge)

Opnfv lab-aaS

Medium
(Access/regional)

Huge
(Global data center)

Deliverables

- A new scenario based on k8s integrating micro-service oriented core services
  - Service mesh
  - Monitoring, logging, tracing
  - Networking (from infrastructure), storage
  - Additional services
- Cloud native CD
- Automated testing of resiliency, scalability …
- Example use cases: e.g. vCPE/uCPE, vIMS, vCDN, …
- Identifying and solving NFV specific problems
- Documentation

Project proposal

- https://wiki.opnfv.org/display/PROJ/Clover
- Initial release: Fraser
- Main contributions to OPNFV
  - Provide a new cloud native micro-service oriented scenario for NFV
  - Provide a set of common services
  - Provide tooling to support cloud native continuous delivery at scale
  - Identify and solve NFV specific issues in upstream
  - NFV use case examples and best practices of developing cloud native VNFs
- Looking for interested contributors, feedbacks, ideas