OPENSTACK

컨테이너와 **OpenStack** 활용하기

종합 리뷰

Contents

1. 소개
2. I. OpenStack에서 컨테이너에 대해 높은 곳에서 바라보기
3. OpenStack 컨테이너 통합 포인트
4. III. 사용 사례
5. IV. 결론
6. V. 오픈소스 프로젝트 인덱스
7. VI. 저자
8. Translators

소개

전체 프라이빗 클라우드 인프라를 처음부터 구축하는 일이 있다고 상상해봅시다. 제한된 예산과 작지만 헌신적인 팀이 있으며 기적을 일으켜달라는 요청을 받았습니다.

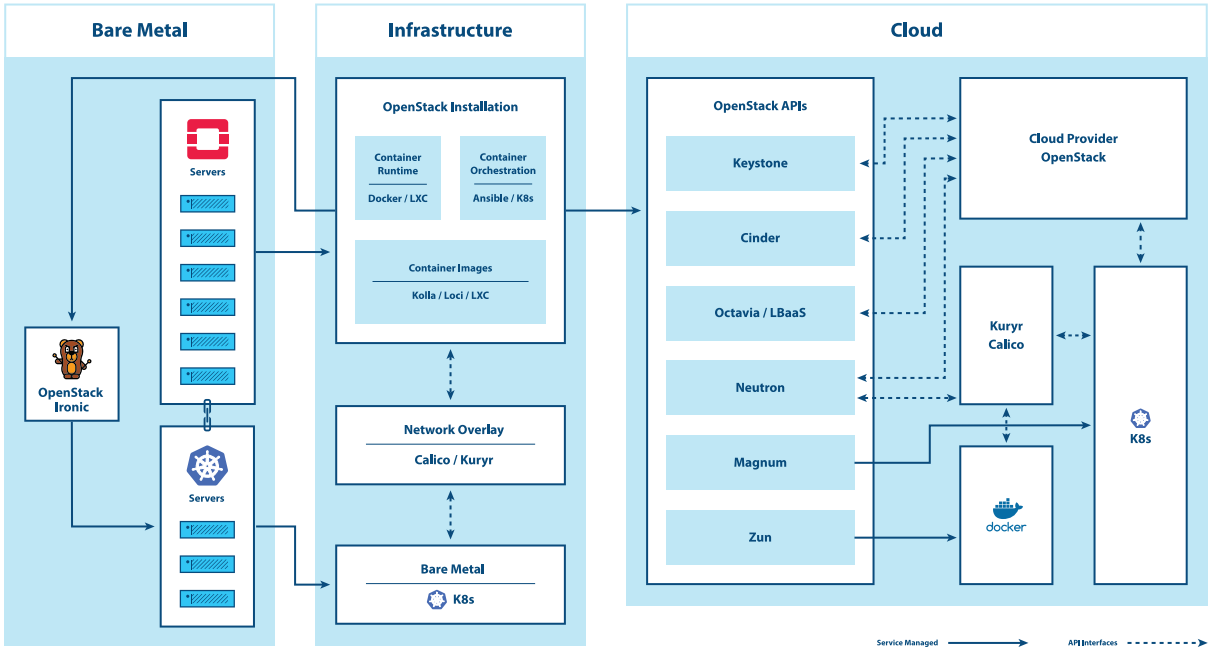
몇 년전, 당신은 가상머신에서 애플리케이션이 작동하도록 인프라를 구축하였습니다. 그리고 일부 레거시 어플리케이션은 베어 메탈(bare-metal)에서 작동하도록 구축하였습니다. 인프라가 발전함에 따라, 가상 머신(VM)은 더 높은 수준의 효율성과 민첩성을 가지게되었지만, VM 만으론 애플리케이션 배포에 대한 민첩한 접근 방식의 요구 사항을 완전히 충족 시키지 못했습니다. 여전히 VM이 작동중인 많은 애플리케이션에서 사용하는 토대가 되고 있지만, 점점 개발자들은 최첨단 애플리케이션 배포에 있어서 컨테이너에 대한 최신의 기술 트렌드를 찾아갈것이고 컨테이너의 민첩성과 효율성이 향상될 것으로 보고 있습니다.

Docker와 Kubernetes같은 컨테이너 기술은 컨테이너화된 애플리케이션을 빌드하는데 있어 최고의 표준이 되고 있습니다. 이 기술들은 개발 민첩성을 제한하는 복잡성으로부터 자유로운 조직을 만드는데 도움을 줍니다. 컨테이너, 컨테이너 인프라, 컨테이너 배포기술은 다양한 사용 사례에 적용 할 수 있을 정도로 매우 강력한 추상화로 입증되었습니다. Kubernetes 같은 것을 사용하여 조직에서 애플리케이션 딜리버리용 컨테이너를 배포할 수 있는 전용 클라우드를 제공할 수 있습니다.

그러나 최첨단 프라이빗 클라우드는 컨테이너만을 위한 것이 아니며, 컨테이너는 모든 워크 로드와 사용 사례에 대한 적합한 기술이 아닙니다. 오늘날 대부분의 프라이빗 클라우드 인프라는 인프라 관리를 위한 베어 메탈 (bare-metal) 머신, 레거시 애플리케이션을위한 가상머신, 새로운 애플리케이션을 위한 컨테이너를 포괄하여야 합니다. 이 세가지 접근 방식을 모두 지원하고 관리, 조율하는 능력이 운영 효율성의 핵심입니다.

OpenStack은 현재 프라이빗 클라우드를 구축할때 네트워크, 스토리지, 컴퓨팅 인프라를 관리하는 기능과 가상 머신, 베어 메탈(bare-metal), 컨테이너를 하나의 컨트롤 플레인에서 제공하는 최상의 옵션입니다. Kubernetes 는 가장 인기있는 컨테이너 오케스트레이터이며 견고한 클라우드 인프라의 가용성을 통해 애플리케이션 딜리버리에 대한 방법을 변경하였으며, OpenStack은 호스팅 애플리케이션을 위한 가장 포괄적인 오픈소스 인프라를 제공합니다. OpenStack의 멀티 테넌트 (multi-tenant) 클라우드 인프라는 Kubernetes 에 자연스럽게 연결되어 여러 통합 포인트, 배포 솔루션, 여러 클라우드와 연합하여 사용할 수 있는 기능이 있습니다.

이 백서는 OpenStack에서 컨테이너가 작동하는 방식을 탐구하고, 다양한 사용 사례를 검토하고, OpenStack과 기타 다른 곳에서 컨테이너를 쉽게 태책하고 활용할 수 있는 기술을 만드는데 도움이되는 오픈소스 프로젝트 개요를 제공합니다.



I. OpenStack에서 컨테이너에 대해 높은 곳에서 바라보기

컨테이너와 OpenStack이 교차하는 세가지 주요 시나리오가 있습니다.

Infrastructure containers라고 하는 첫 번째 시나리오를 통해서 운영자는 클라우드 인프라의 배포, 관리 및 운영을 향상시키는 방식으로 컨테이너를 활용할 수 있습니다. 이 시나리오에서 컨테이너는 베어-메탈 인프라에 설치되며 호스트 리소스에 대해 액세스 권한이 주어집니다. 이 액세스를 통해 컨테이너 런타임 동안 일반적으로는 사용자가 알지 못하게 숨기려고 하는 컴퓨팅, 네트워킹 및 저장소 자원을 직접 활용할 수 있습니다. 컨테이너는 인프라 애플리케이션이 기본 시스템 자원을 직접 관리, 조작할 수 있도록 하면서 각 애플리케이션이 의존하는 복잡한 의존성 집합을 격리합니다. 서비스를 업그레이드 할 때가 되면, 같은 위치의 서비스를 방해하는 의존성을 변경하지 않고도 업그레이드를 처리할 수 있습니다.

OpenStack의 최신 버전은 이 Infrastructure container 모델을 채택했으며, 현재는 오케스트레이션 도구와 컨테이너화된 서비스를 결합한 OpenStack 배포의 전체 라이프 사이클을 관리하는 것이 일반적입니다. Infrastructure containers를 사용하면 운영자는 컨테이너 오케스트레이션 기술을 통해 많은 문제들을 해결할 수 있습니다. 특히 OpenStack을 비롯한 기존의 소프트웨어의 신속한 반복 작업 / 업그레이드 해야하는 문제를 해결할 수 있습니다. 컨테이너 내에서 OpenStack을 운용하면 운영자는 서비스의 새로운 구성 요소 추가, 소프트웨어 버전의 신속한 업그레이드, 그리고 컴퓨터 및 데이터 센터간의 신속한 rolling 업데이트를 포함한 Day 2 Challenge를 해결할 수 있게 됩니다. 이러한 접근법은 OpenStack의 배포와 업그레이드 문제에 컨테이너가 갖는 민첩성을 가져다 줍니다.

두 번째 시나리오는 클라우드 인프라에서 컨테이너화된 응용 프로그램 프레임워크를 호스팅하는 것과 관련됩니다. 이는 Docker Swarm과 Kubernetes와 같은 컨테이너 오케스트레이션 엔진(COEs)이나 경량 컨테이너 중심 서비스와 무서버 애플리케이션 프로그래밍 인터페이스(APIs)를 포함할 수 있습니다. bare-metal이든 VM이든 OpenStack 커뮤니티는 tenant 격리된 클라우드 호스트에 컨테이너화된 응용 프로그램들을 안전하게 제공이 가능하도록 보장하기 위해 노력해왔습니다. 이 시나리오는 Kubernetes와 같은 프로젝트가 스토리지, 로드밸런싱, 그리고 식별을 위해 OpenStack API를 직접 활용할 수 있게 해주는 드라이버에 의해

촉진됩니다. 이는 또한 Kubernetes로 관리되는 클러스터와 필요한 응용 프로그램 컨테이너의 프로비저닝을 위한 API도 포함합니다. 이러한 기능을 통해, 개발팀은 새로운 컨테이너 응용 프로그램을 작성하고 빠르게 OpenStack 클라우드에 Kubernetes 클러스터들을 프로비저닝할 수 있습니다. 이는 응용 프로그램을 제품에 배포하기 위한 강력한 자동화로, 코드의 개발과 테스트, 그리고 디버깅에 필요한 자원들을 제공하는 완벽한 응용 프로그램 생명주기 솔루션입니다.

마지막 시나리오에서는 독립 OpenStack과 COE 배포 사이의 상호 작용을 고려하고, 이 백서에서는 특히 Kubernetes 클러스터를 고려합니다. OpenStack과 Kubernetes 클러스터 모두에서의 API의 일관성과 상호운용성은 이 시나리오를 위한 성공의 주요 원천입니다. 예를 들어, Kubernetes가 OpenStack Cinder에 호스팅된 볼륨에 OpenStack Keystone을 인증과 인증 백엔드로 사용하여 직접 연결하거나 OpenStack Neutron을 오버레이 네트워크로 OpenStack Kuryr과 연결할 수 있습니다. 반대로, OpenStack 클라우드는 Calico와 같은 프로젝트를 위한 Neutron 드라이버로 같은 오버레이 네트워크를 Kubernetes 클러스터로서 공유하는 것이 가능합니다. 세 번째 시나리오는 클라우드 서비스의 호스팅 방식(Kubernetes나 OpenStack)에 초점을 두지 않고 독립 서비스의 상호작용 방식에 초점을 둡니다.

OpenStack 컨테이너 통합 포인트

OpenStack 인프라를 컨테이너에 배포하기

도입부에서 언급하였듯이, 컨테이너가 인프라 코드 관리에 대한 새로운 접근 방법을 열어주었던 이유로 인해 OpenStack에 대한 배포 및 관리 방식은 컨테이너의 등장과 함께 큰 변화를 이루었습니다. 이전 관리 전략에서는 무거운 골든 머신 이미지 생성 및 유지 보수 또는 깨지기 쉬운 상태 관리, 구성 관리 시스템 사용을 필요로 하였습니다. 각 접근 방식에는 복잡성과 제한 사항이 수반됩니다. 난이도를 추가하는 것은 릴리즈에 따라 변경이 이루어지는 각각의 의존성을 필요로 하는 서비스 집합에 대한 관리에 해당합니다. 응용 프로그램을 격리하는 어떤 형태가 없다면, 해당 의존성을 해결하는 것은 불가능하지는 않더라도 어려워집니다.

인프라 컨테이너는 새로운 OpenStack 배포 프로젝트를 의존성 문제를 우아하게 해결하는 동시에 해당 두 가지 사이에서의 균형을 흔드는 것을 가능하게 만듭니다. 가볍고, 독립적이면서, 자신을 포함하며 보통 상태가 없는 응용 프로그램 컨테이너를 사용하여, 클라우드 운영자는 복잡한 제어 계층을 배포할 때 엄청난 유연성을 얻습니다. 컨테이너 런타임 및 오케스트레이션 엔진을 결합함으로써 인프라 컨테이너는 복잡하고 고가용성이 있는 인프라를 쉽게 배포, 관리 및 업그레이드 가능하게 만들어 줍니다.

OpenStack 클러스터를 만드는데 있어, 배포 기술을 선택하는 것에 대해 몇 가지 관점이 있습니다. 운영자는 Linux 컨테이너또는 Docker를 기반 컨테이너로 선택하고, 응용 프로그램 컨테이너를 사전에 빌드하거나 사용자 정의 빌드를 하여, 오케스트레이션을 위해 전통적인 구성 관리 시스템 또는 Kubernetes와 같은 보다 현대적인 접근 방식을 선택할 수 있습니다. 표 1은 기존 OpenStack 배포 프로젝트 및 각 프로젝트에 기반한 기술을 요약한 것입니다.

Project	Container Type	Supported Containers	Project
OpenStack-Ansible	LXC	OSA LXC Containers	Ansible
Kolla-Ansible	Docker	Kolla Containers	Ansible
Triple-O	Docker	Kolla Containers	Ansible
OpenStack-Helm	Docker	Kolla Containers Loci Containers	Kubernetes and Helm

Table 1

기반에 놓여 있는 각 해당 배포 시스템들은 OpenStack 코드 및 지원 서비스를 위해 컨테이너 집합을 만드는 데 있어 서로 다른 접근 방식에 해당합니다. OpenStack Ansible (OSA) 및 Kolla 프로젝트는 자체 프로젝트에 호스팅이 이루어지는 빌드 시스템을 사용하는 반면, LOCI는 특정 오케스트레이션 시스템을 염두해 두지 않으면서 프로젝트 응용 프로그램 컨테이너를 만드는 것에 집중합니다. 높은 레벨에서 보면, 차이점은 다음과 같습니다:

1. OSA는 저수준 LXC 컨테이너에 의존한다는 점에서 독특하며, LXC 애플리케이션 컨테이너를 생성하기 위한 커스텀 빌드 시스템을 갖추고 있습니다.
2. Kolla 빌드 시스템은 OpenStack 배포를 초기화 및 관리하는 보조 컨테이너와 함께 각 서비스에 대해 하나씩 Docker 컨테이너를 만들어 냅니다. Kolla 컨테이너는 기반 운영체제 선택, 소스 또는 패키지 설치, 그리고 심지어 이후 커스터마이징에 대한 템플릿 엔진과 함께 쉽게 구성이 가능합니다.
3. OpenStack 응용 프로그램 컨테이너를 만드는 데 있어 마지막 옵션은 LOCI입니다. LOCI 역시 Docker 컨테이너를 만들고, 각 프로젝트마다 한 개의 컨테이너를 전달합니다. LOCI는 모든 공통 배포에 있어 작고 안전한 컨테이너를 빠르게 만드는 데 초점을 두고 있습니다. 단, 컨테이너들은 배포 시스템에 의해 빌드가 이루어지는 데 있어 기반을 두는데 사용되어진다는 것을 전제로 합니다.

Bare Metal Infrastructure - OpenStack 그리고 부트스트랩 문제 해결

모든 클라우드의 기반에서 보면, 인프라 서비스를 호스팅하는 베어메탈 서버에 대한 데이터센터가 있습니다. 심지어 "서버리스 컴퓨팅" 도 데이터센터 내 하드웨어 상에서 클라우드를 통해 소프트웨어를 실행합니다. 하드웨어 인프라를 어떻게 부트스트래핑할지에 대한 문제는 OpenStack 소프트웨어가 베어메탈 관리를 클라우드와 같은 퀄리티로 제공하는 방식을 통해 유일하게 해결하는 치명적인 문제입니다.

OpenStack Ironic은 bare-metal as a service를 제공합니다. 단독 서비스로서 베어메탈 노드를 발견 가능하고, 관리 데이터베이스를 이들을 카탈로그화하며, 등록, 프로비저닝, 관리, 그리고 폐기를 포함하는 전체 서버 라이프사이클을 관리할 수 있습니다. OpenStack Nova와 연결이 이루어지도록 사용을 하고 OpenStack 서비스 전체 집합과 함께 결합한다면, 전체 베어메탈 인프라를 관리하는데 있어 매우 강력하고, 클라우드와 같은 서비스를 제공합니다.

이는 다음 질문을 불러옵니다: 어떻게 하나의 부트스트랩 OpenStack이 베어메탈 인프라를 관리하기 위해 서비스가 이루어질까요? 일반적인 해결책 중 하나는 시드 설치를 생성하기 위해 이전 섹션에서 언급된 것과 동일한 컨테이너 기반 설치 도구를 사용하는 것입니다. '언더클라우드' 라 불리는 해당 시드는 마치 가상화된 클라우드인 것처럼 베어메탈 클러스터 관리를 완전하게 자동화하는 데 사용할 수 있습니다.

이는 OpenStack 가상화가 단지 베어메탈 클라우드 위에서 실행 뿐만 아니라, 인증, 스토리지, 네트워킹, 그리고 OpenStack 서비스를 통해 사용 가능한 다른 클라우드 API 모두의 장점을 이용할 수 있는 Kubernetes만 설치된 베어메탈을 실행하는기회를 여는 것입니다.

OpenStack 상에서 컨테이너 기반 애플리케이션 제공

인프라 컨테이너 및 베어메탈 인프라 모두 중요하지만, 대부분의 사람들은 컨테이너를, 응용 프로그램 컨테이너로 간주한다고 생각합니다. 컨테이너가 제공해 주는 격리, 캡슐화, 그리고 관리 용이성은 응용 프로그램을 전달함에 있어 이상적인 솔루션으로 만들어줍니다. 그러나 컨테이너들은 베어메탈, 공용 클라우드, 또는 사설 클라우드 상관없이 그들의 형태를 제공하는 호스트 플랫폼을 여전히 필요로 합니다.

Kubernetes는 애플리케이션을 제공하기 위한 플랫폼으로, 영구 스토리지, 로드 밸런서, 네트워크 및 컴퓨트 노드의 동적 할당과 같은 중요한 인프라 제공을 자동화 할 수 있는 클라우드 API를 통해서 최고의 성능을 발휘합니다. OpenStack은 온 프레미스 private 클라우드 또는 사용 가능한 public 또는 managed OpenStack 클라우드를 통해 클라우드 인프라를 제공합니다.

OpenStack은 "Kubernetes/Cloud Provider OpenStack" 플러그인을 유지보수하는 데 있어 활발한 개발자 팀과 함께 Kubernetes에 대한 첫 업스트림 클라우드 제공자 중 하나였습니다. 이 플러그인은 Kubernetes가 Cinder 블록 스토리지, Neutron 및 Octavia로드밸런서, 그리고 Nova를 사용한 compute 자원에 대한 직접적인 관리를 가능하게 해줍니다. 동일한 제공자를 사용하는 것은 Kubernetes 설치, 드라이버 로딩을 위한 플래그 설정, 그리고 로컬 사용자 클라우드 자격증명을 제공하는 데 있어 드라이버를 배포하는 것과 같이 간단하게 만들어 줍니다.

Kubernetes 및 다른 응용 프로그램 프레임워크를 OpenStack 위에 설치하기 위한 여러 솔루션이 있습니다. 컨테이너 프레임워크를 제공하는 쉬운 방법 중 하나는 Kubernetes를 포함하여 여러 응용 프로그램 플랫폼 중 하나를 선택하여 완전하게 관리가 이루어지는 클러스터를 만드는 간단한 API를 제공하는 Magnum을 사용하는 것입니다. 이것은 OpenStack API 및 클라우드 제공자 플러그인에 의존적인 Kubernetes 배포 시스템에 대한 예시입니다. 예를 들어, 지금 현재 파트너사 클라우드 뿐만 아니라 200개가 넘는 독립적이고 페더레이션이 이루어진 CERN OpenStack 온사이트 클라우드에서의 Kubernetes 설치에서 사용이 이루어지고 있습니다. 선호하는 OpenStack 클라우드 내에서 사용 가능한 Magnum API를 가지고 있지 않은 경우에는, 대상 Kubernetes 클러스터를 OpenStack에서 설치 및 관리를 하기 위해 kubeadm, Kubernetes Anywhere, Cross-Cloud, 또는 Kubespray와 같은 Kubernetes 설치 도구를 사용할 수 있습니다. 각각 표준 Kubernetes를 사용하기에, 클라우드 제공자 인터페이스가 스토리지 및 로드밸런싱을 이용하기 위한 클라우드 프로바이더 인터페이스를 활성화하는 것이 쉽습니다.

또 다른 OpenStack 프로젝트인 Zun은 서버 또는 클러스터 관리를 필요로 하지 않으면서 개별 클러스터를 관리하기 위해 가벼운 컨테이너 서비스 API를 제공합니다. OpenStack에 의해 호스팅이 이루어지는 Kubernetes 클러스터는 Nova API를 통해 직접적으로 클러스터에 클라우드 자원을 동적으로 추가 또는 제거할 수 있기에 탄력적입니다. 다른 방식으로는, Kubernetes가 pod 인프라에 대한 관리를 Zun으로 전환하면서 컨테이너 백엔드를 OpenStack Zun에 제공할 수 있습니다. 직접적으로 서버를 생성하는 데 사용하지 않는 매우 가볍고 멀티테넌트 서비스 API를 제공합니다. Neutron 및 Cinder와의 직접적인 통합은 개별 컨테이너를 위해 네트워킹 및 볼륨을 제공하는 데 사용됩니다.

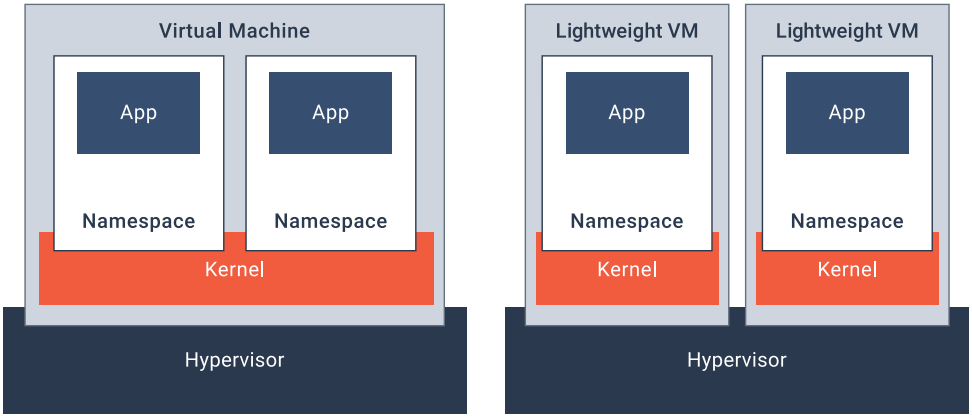
마지막으로, Qinling 프로젝트는 Lambda, Azure Function, 또는 Google Cloud Functions와 유사한 서버리스 펑션을 지원하는 플랫폼을 제공하는 것을 목표로 합니다. 게다가 컨테이너 관리를 추상화하고, 사용자들이 필요로 할 때 확장이 가능한, 이벤트 주도 및 서버리스 컨퓨팅 경험과 함께 사용자들이 개발을 가속화하도록 도와줍니다. Qinling은 Kubernetes, Docker Swarm, 그리고 로컬 스토리지 및 다양한 OpenStack Swift와 같은 다양한 대중적인 펑션 패키지 스토리지에 대한 서로 다른 컨테이너 오케스트레이션 백엔드를 지원합니다.

Kata Container - 가상화를 통한 애플리케이션 보안

Kata Containers는 새로운 오픈 소스 프로젝트로, 컨테이너 생태계 내에서 완벽하게 통합되는 경량 가상 머신의 새로운 구현물입니다. Kata Container는 컨테이너처럼 가볍고 빠르며 Docker 및 Kubernetes (k8s)와 같은 인기있는 오케스트레이션 도구를 비롯한 컨테이너 관리 레이어와 통합되는 동시에 VM의 보안 이점도 제공합니다. Kata Containers는 OpenStack Foundation이 활발하게 활동하고있는 Open Container Initiative (OCI)표준을 준수합니다. Kata Containers는 OpenStack Foundation에서 호스팅되지만 자체 관리 및 커뮤니티가 있는 OpenStack 프로젝트와는 별도의 프로젝트입니다.

컨테이너로의 업계 이동은 신뢰할 수있는 작업 부하와 신뢰할 수없는 작업 부하가 혼합된 멀티 테넌트 환경에서 사용자 작업 부하를 보호하는데 있어 고유한 문제를 제시합니다. Kata Containers는 하드웨어 기반 분리를 각 컨테이너 또는 컨테이너 모음의 경계로 사용합니다. 이 접근 방식은 전통적인 컨테이너 아키텍처에서 공유 커널의 보안 문제를 해결합니다.

Kata Containers는 지속적인 통합 / 지속적인 배포와 같은 주문형, 이벤트 기반 배포 및 장기간 실행되는 웹 서버 응용 프로그램 모두에 매우 적합합니다. Kata는 기존 guest 커널 및 device pass through 기능을 지원하므로 전통적인 가상화 환경에서 컨테이너로 쉽게 전환 할 수 있습니다. Kata Containers는 향상된 보안, 확장 성 및 높은 리소스 활용률을 제공함과 동시에 전체적인 단순화 된 스택을 제공합니다.



Containers in Cloud Today
(Shared kernel, isolation within namespace)

Kata Containers
A lightweight virtual machine isolates each container/pod and provides a separate kernel for each container/pod.

Side-by-Side OpenStack과 Kubernetes의 통합

오픈 소스 플랫폼을 선택할 때의 주요 이점 중 하나는 해당 플랫폼의 표준 배포 전반에 걸친 인터페이스의 안정성입니다. OpenStack Foundation 및 CNCF(Cloud Native Computing Foundation)는 OpenStack 클라우드 및 Kubernetes 클러스터에 대한 상호 운용성 표준을 유지하므로 라이브러리, 응용 프로그램 및 드라이버가 배포 위치와 상관없이 모든 플랫폼에서 작동합니다. 이는 OpenStack과 Kubernetes가 다른 쪽에서 제공하는 리소스를 활용할 수 있도록 나란히 통합할 기회를 만듭니다.

Kubernetes 커뮤니티의 OpenStack Special Interest Group(SIG-OpenStack)은 Cloud Provider OpenStack 플러그인을 관리합니다. OpenStack에서 Kubernetes를 실행하기위한 클라우드 공급자 인터페이스 외에도 Kubernetes가 개별 OpenStack 서비스를 이용할 수있게 해주는 몇 가지 드라이버를 유지 관리합니다. 이 드라이버에 포함된 것은 다음과 같습니다.

- 두 개의 독립 실행형 Cinder 드라이버. Flex Volume 드라이버는 드라이버와 인터페이스하기 위해 exec 기반 모델을 사용하고 컨테이너 오케스트레이션 시스템에 대한 표준 인터페이스를 사용하여 컨테이너 작업 부하에 임의의 스토리지 시스템을 노출시키는 CSI(Container Storage Interface) 드라이버를 사용합니다. 70 개가 넘는 스토리지 드라이버를 지원하는이 드라이버는 단일 Cinder API를 통해 검증된 테스트를 거친 오픈소스 저장 장치와 풍부한 인터페이스를 가능하게 합니다.
- webhook 기반의 인증과 인가는 Keystone에 연결됩니다. 인증과 인가, 각 모드는 서로 독립적으로 구성 할 수 있습니다. 현재 진행중인 작업이지만 이 인터페이스는 OpenStack Keystone으로 Kubernetes RBAC를 구현하는 소프트 멀티테넌시를 지원합니다.

OpenStack과 Kubernetes는 다양한 드라이버로 뒷받침되는 고도로 역동적인 네트워킹 모델을 지원합니다. 이러한 표준 네트워크 인터페이스 덕분에 강력한 네트워크 통합을 통해 독립형 OpenStack 및 Kubernetes 클러스터를 쉽게 구축 할 수 있습니다. OpenStack에서 Kuryr 프로젝트는 Neutron 네트워킹을 Docker 및 Kubernetes에 전달하는 CNI(Common Network Interface) 드라이버를 생성합니다. 한편, Calico가 제공하는 Neutron 드라이버와 같은 프로젝트를 통해 표준 Neutron API를 통해 인기있는 Kubernetes 네트워크 오버레이에 직접 액세스할 수 있습니다.

III. 사용 사례

OpenStack 커뮤니티의 많은 구성원들이 컨테이너와 관련된 다양한 OpenStack 프로젝트에 새로운 코드를 제공하고 있으며, 컨테이너의 의미와 이점을 평가하고, 생산 과정에서 컨테이너를 사용하여 문제를 해결하고 새로운 기능을 활용할 수 있도록 합니다. 이 섹션에서는 가장 흥미로운 사례 연구 중 일부를 중점적으로 설명합니다.

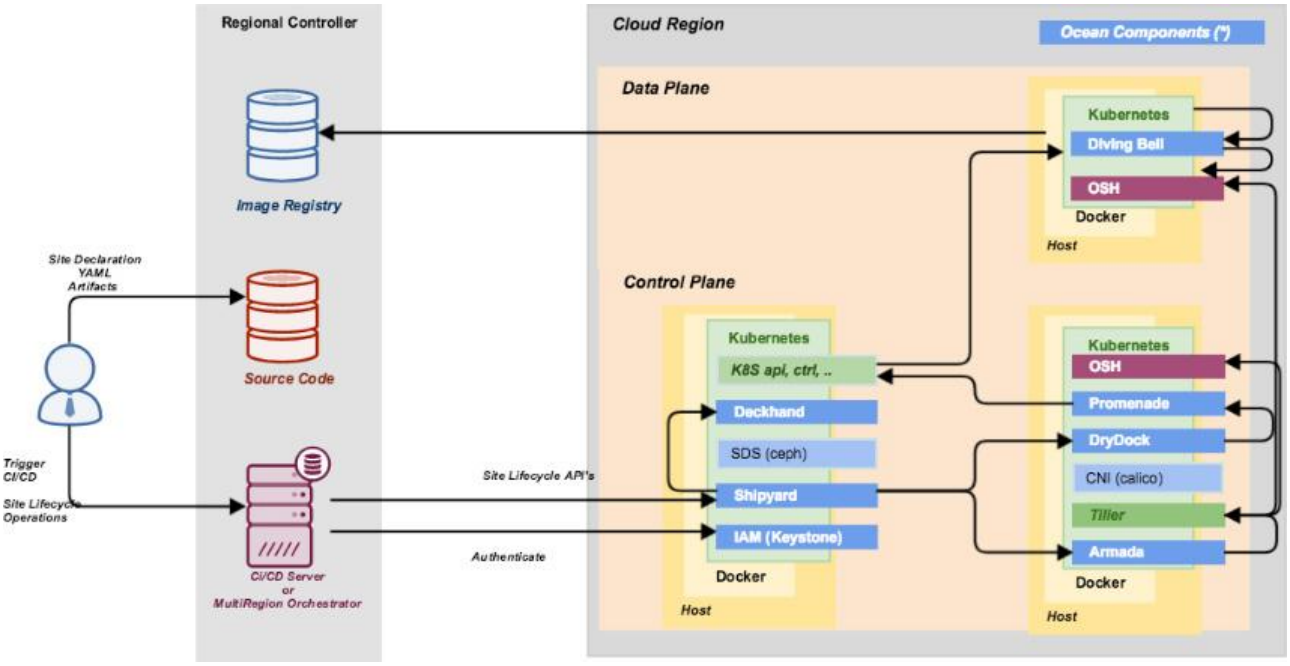
Select a Case Study

- AT&T
- Cern
- SK 텔레콤
- Superfluidity

AT&T

세계 최대의 통신 회사 중 하나인 AT & T는 컨테이너 기술을 활용하여 OpenStack 자체를 배포 및 관리하고 인프라 스트럭처 컨테이너를 사용하여 컨테이너 및 OpenStack에 5G 인프라를 구축하는 단순성과 효율성을 제공합니다.

그들의 목표를 달성하기 위해 AT&T는 OpenStack-Helm 프로젝트를 사용하여 Kubernetes 클러스터에서 LOCI 기반 OpenStack 이미지를 조율하고 Kubernetes, Docker 및 핵심 OpenStack 서비스를 활용합니다. 그들은 또한 Bandit, Tempest, Patrole 및 많은 다른 OpenStack 프로젝트를 사용하고 있습니다. 또한 AT&T는 Airship이라는 undercloud 프로젝트 콜렉션을 소개하기 위해 커뮤니티에서 협력하고 있습니다. 이 프로젝트는 베어메탈로부터 OpenStack 워크로드를 실행하는 production 등급의 Kubernetes로 클라우드를 제공합니다.



AT&T는 컨테이너화를 통해 전통적인 배포 유형 활동을 왼쪽으로 멀리 이동시키고 CI/CD를 사용하여 유효성을 검사할 수 있음을 확인했습니다. Kubernetes는 또한 대규모 확장성과 탄력성을 제공할 뿐 아니라 OpenStack-Helm이 운영 활동을 선언적으로 구성하고, 설정을 추가하고, 테넌트 워크로드에 영향 없이 롤링 업그레이드와 업데이트를 수행할 수있게 해주는 후크를 제공합니다.

컨테이너 기술을 활용하여 OpenStack을 배포하고 관리하는 경우, 탄력성이 뛰어난 플랫폼을 사용하고 더 잦은 중단 없이 클라우드 기능을 사용할 수 있다는 점을 제외하면 테넌트에게 미치는 영향은 분명하지 않습니다. AT&T의 운영팀은 새로운 경험을 통해 사이트에 대한 선언적 구성을 정의하고 Kubernetes 지향 자동화가 배포를 직접 수행할 수 있도록 더 많은 노력을 기울일 것입니다.

AT&T는 이러한 아키텍처를 사용하여 소비자 및 비즈니스 중심의 제품과 서비스의 백본을 형성하는 가상 네트워크 기능에 전력을 공급하는 것을 목표로 합니다. AT&T의 컨테이너화된 Network Cloud의 초기 사용 사례는 새로운 5G 네트워킹을 위한 VNF의 초기 배포일 것입니다. OpenStack은 AT&T의 VNF 중심의 클라우드 사용 사례에 적합하고, 앞으로도 계속 될 것입니다. 컨테이너화는 단순히 AT&T가 OpenStack 인프라를 보다 안정적이고 신속하게, 그리고 zero-touch 방식으로 배포, 관리 및 확장할 수 있도록 해주는 진화된 형태입니다.

운영 측면에서 AT&T는 이러한 접근 방식을 여전히 테스트하고 있지만, 연말 전에 5G 서비스를 production에 투입하기로 약속했습니다. OpenStack과 컨테이너 기술은 AT&T의 수백만 사용자에게 전략적으로 중요한 이 서비스의 중추적인 역할을 형성할 것입니다. 5G 서비스를 배포하면 대량으로 분산된 production 환경에서 OpenStack과 컨테이너의 연관성을 입증할 수 있습니다.

Cern

유럽 연구기구인 CERN은 물리학자와 엔지니어가 우주의 기본 구조를 탐구할 수 있게하며, 세계에서 가장 크고 복잡한 과학 도구를 사용하여 물질의 기본 구성요소인 기본 입자를 연구합니다. CERN 클라우드는 물리학자에게 과학 컴퓨팅을 위한 컴퓨팅 리소스를 제공하고 대형 Hadron Collider에서 오는 데이터 및 기타 실험을 분석합니다.

CERN은 2013년부터 OpenStack을 사용 중이며 현재 단일 클라우드 내에서 가상시스템, 베어메탈 및 컨테이너 서비스를 제공하고 있습니다. 컨테이너는 OpenStack Magnum을 통해 프로비저닝된 사용 사례에 따라 가상시스템 또는 베어메탈에서 실행됩니다. Kubernetes, Docker Swarm 및 DC/OS를 비롯한 여러 가지 다양한 컨테이너 기술을 사용할 수 있습니다.

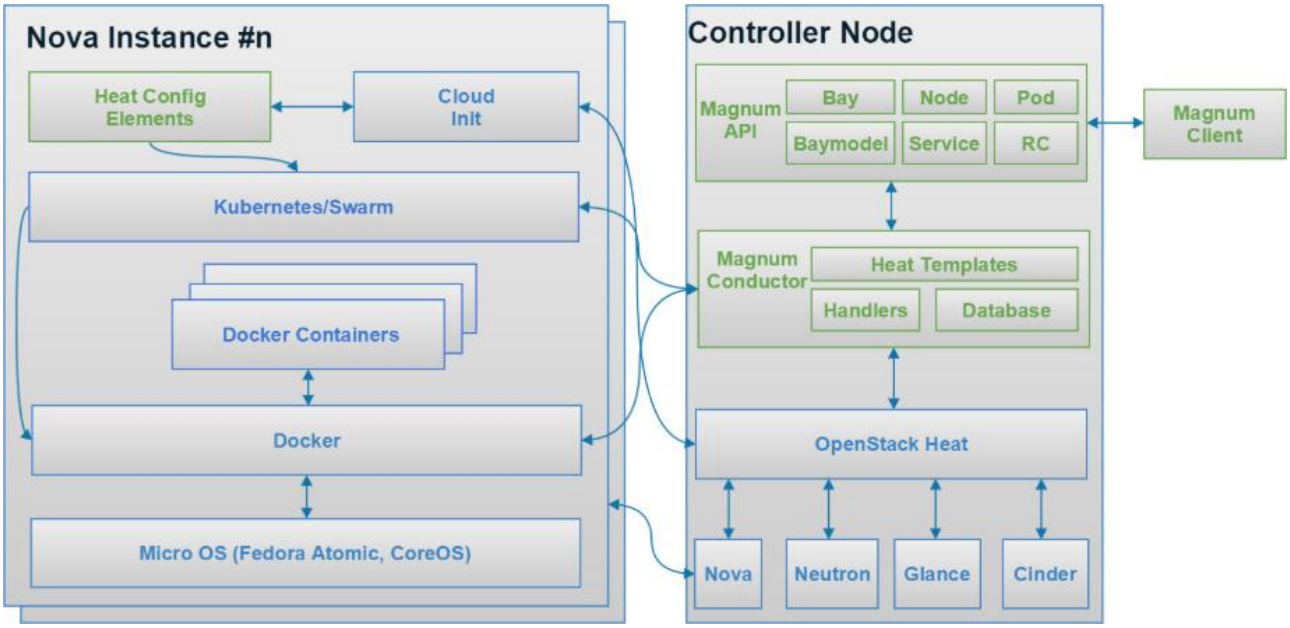
CERN은 현재 OpenStack 위에서 Magnum을 통해 프로비저닝된 250 개의 컨테이너 클러스터를 실행 중입니다.



CERN의 OpenStack 클라우드는 사용자에게 몇 가지 명령 또는 웹 GUI를 통해 구성된 컨테이너 엔진을 요청할 수있는 셀프 서비스 액세스 권한을 제공합니다. 이를 통해 기술을 신속하게 활용할 수 있으며 필요한 경우 1000s 노드까지 확장할 수 있습니다. CERN 스토리지 및 인증 서비스에 내장된 모니터링 및 통합을 통해 모범사례 구성을 사용할 수 있습니다.

이 리소스 풀을 효율적으로 실행하여 추가 작업 인력 필요없이 확장하면 일관된 관리 프로세스와 도구가 필요합니다. Magnum을 통해 OpenStack에 컨테이너를 추가하는 것은 하드웨어 복구 프로세스 및 일관된 권한 부여 모델과 같이 이전에 개발된 자동화를 사용하도록 만들었고, 또한 사용자 요구에 따라 리소스를 신속하게 재할당할 수 있게 되었습니다.

공개적으로 자금 지원을 받는 연구실로서, Kubernetes 및 OpenStack과 같은 오픈소스 솔루션은 다른 조직과 공동 작업을 수행하고 커뮤니티에 제공할 수있는 프레임워크를 제공합니다. CERN은 Rackspace 및 Huawei와 같은 CERN openlab framework를 통해 다수의 공급업체와 협력하여 Magnum 및 Federation과 같은 기능을 대규모로 제공합니다. 이러한 경험은 OpenStack Special Interest Groups, Square Kilometer Array(SKA)와 같은 다른 science, Kubecon Europe과 같은 public presentations 및 OpenStack in Production과 같은 블로그에 공유합니다.



CERN에서 Magnum에 의해 제공되는 컨테이너에서 실행되는 여러 작업 부하는 다음을 포함합니다.

- Reana/Recast
  - 이러한 도구들은 High Energy Physics에서 재사용 가능한 워크플로우를 실행하기 위한 프레임워크를 제공합니다. 컨테이너는 분석 소프트웨어와 데이터를 한번에 쉽게 공유할 수있는 장치로 패키지화할 수있을 뿐만 아니라 온프레미스와 외부 리소스의 사용을 쉽게 확장할 수 있습니다. 작업은 분석 및 데이터 보존 활동을 지원하는 Yadage Workflows를 기반으로하는 Kubernetes 작업으로 스케줄링됩니다.
- Spark as a Service
  - 최근에 Kubernetes는 Spark를 위한 resource manager로 추가되었습니다. 스파크(Spark)는 운전자와 집행자를 포드(pod)로 만들고 Kubernetes는 스케줄링과 라이프 사이클을 담당합니다. CERN의 IT 부서는 사용자가 OpenStack Magnum을 사용하여 필요에 따라 Kubernetes 클러스터를 만들고 Kubernetes에 Spark를 배포하여 CERN의 특수화된 파일 시스템 및 데이터 소스에 필요한 모든 요구된 통합 내용을 안전하게 제공할 수있는 서비스를 개발하고 있습니다. 사용자는 몇 가지 명령을 사용하여 필요할 때만 원하는 크기로 Spark 배포를 효과적으로 만들 수 있으며 실행 중에 배포를 확장 또는 축소 할 수 있습니다.
- LHC 업그레이드를 위한 LHC experiment detector trigger simulation
  - LHC는 collision을 필터링하는 experiment trigger farm에서 상당한 향상을 요구하는 2020년 동안 더 높은 luminosity로 업그레이드될 예정입니다. 대규모 Kubernetes 클러스터는 ATLAS실험을 위한 다양한 접근법을 시뮬레이션하고 디자인을 검증하여 Kubernetes 및 OpenStack 구성 요소에 대한 몇몇 fine tuning을 가능하게 하기 위해 만들어졌습니다.
- Gitlab 지속적인 통합 Runners
  - 사용자는 Gitlab을 사용하여 CI/CD 작업을 빌드할 수 있고, 공유된 환경 또는 개별 프로젝트 환경에서 그러한 작업을 실행할 수 있습니다. CERN 사용자는 CERN Container Service를 활용하여 소프트웨어 테스트 및 빌드, 컨테이너 이미지 및 문서 작성 및 게시, 자동화된 배포를 비롯한 전체 응용 프로그램 수명주기를 관리하는 복잡한 파이프 라인을 다양한 환경에 설정할 수 있습니다.
- 페더레이션 된 Kubernetes는 외부 클라우드가 있는 팜(farm)을 계산합니다.
  - CERN은 다중 클라우드 운영을 지원하기 위해 Kubernetes 클러스터의 federation 기능을 사용합니다. 다중 클러스터들은 CERN and the T-Systems Open Telekom Cloud와 같은

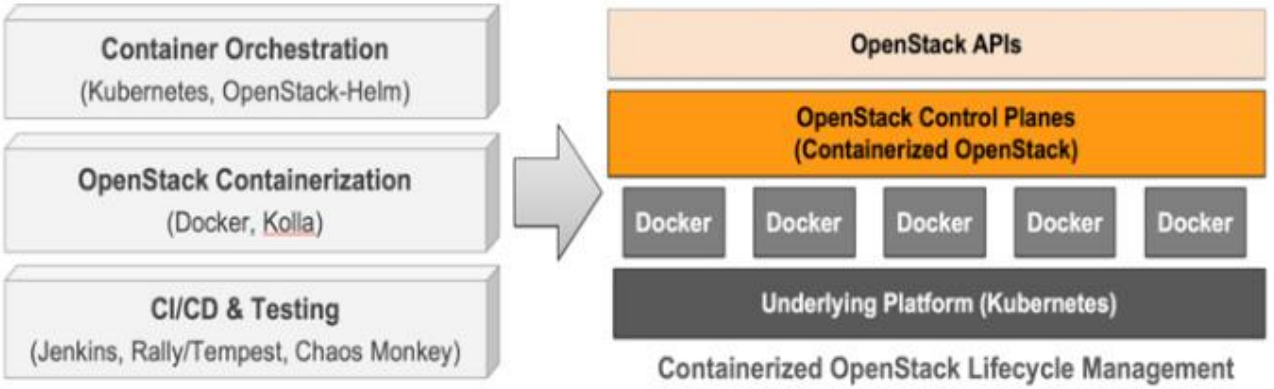
AWS, GCE 및 OpenStack 클라우드를 포함하여, 다양한 기술의 클라우드에 원활히 통합될 수 있습니다. 참고적으로, T-Systems Open Telekom Cloud는 Kubecon 2018에서 시연되었습니다.

가상머신, 컨테이너 엔진 및 베어 메탈을 단일 프레임워크에 통합하면 사용 통계, 소유권 및 할당량을 쉽게 볼 수 있습니다. Kubernetes를 위한 Manila storage driver는 파일 공유에 대한 프로비저닝 투명성을 허용합니다. 이는 용량 계획(capacity planning)을 수행하는 IT 부서와 작업그룹의 우선 순위를 정의하는 실험 리소스 코디네이터 모두를 지원합니다. 직원의 퇴사시의 자원 재할당 또는 자원 만료와 같은 자원 관리 정책은 일관된 워크플로우에서 처리됩니다.

SK 텔레콤

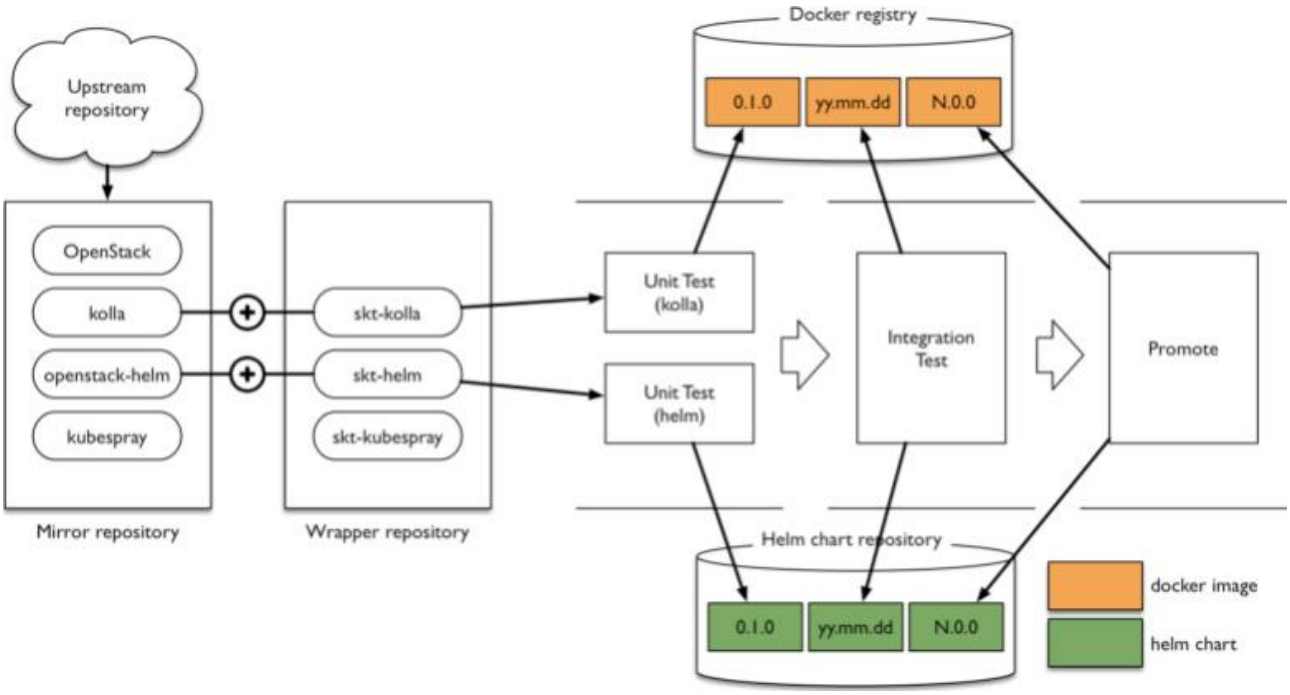
SK Telecom (SKT), South Korea's largest telecommunications operator, has been exploring optimized approaches for deploying OpenStack on Kubernetes with the aim of putting core business functions on containerized OpenStack by the end of 2018. SKT leverages Kolla and Openstack-Helm. with deployments automated by Kubespray. SKT devotes nearly 100% of it's development efforts to OpenStack-Helm, and works closely with AT&T to make OpenStack-Helm successful.

SKT는 또한 Kubernetes의 노력에 대한 OpenStack에 다른 도구를 통합했습니다. 로깅, 모니터링 및 경보의 경우 Prometheus및 Elasticsearch, Fluent-bit및 Kibana, 이들은 모두 OpenStack-Helm 커뮤니티의 기본 레퍼런스 툴입니다. SKT는 이 모든 것을 TACO : SKT All Container OpenStack이라는 단일 폐쇄형 통합 솔루션으로 결합합니다.



SKT는 특히 Kubernetes의 Openstack 컨테이너에 자동화된 CI/CD(continuous continuous/continuous delivery) 파이프라인을 강조합니다. SKT의 CI시스템은 Jenkins, Rally, Tempest, Docker Registry, Jira 및 Bitbucket 등이 있습니다. SKT는 또한 CI 파이프라인을 수행하는 Kubernetes 배포를 위한 탄력성(resiliency) 테스트 도구와 같은 chaos-monkey인 Cookiemonster라는 오픈소스 도구를 개발하여 탄력성 테스트를 수행했습니다.

SKT는 모든 변화에 따라 OpenStack 컨테이너와 Helm charts를 자동으로 빌드하고 테스트합니다. 매일 세 개의 control 노드와 두 개의 compute 노드가 있는 고가용성 OpenStack 배포를 설치하고 Tempest에서 400개의 테스트 케이스를 실행하여 서비스의 유효성을 검사하고 마지막으로 Cookiemonster와 Rally를 이용하여 탄력성(resiliency) 테스트를 수행합니다. 전체 CI 시스템이 다음 다이어그램에 표현되어 있습니다.



SKT는 Airship의 하위 프로젝트인 Armada를 사용하여 배포를 자동화합니다. Airship은 AT&T의 새로운 개방형 인프라 프로젝트로 커뮤니티에 소개되었습니다. SKT는 production 기반의 프로젝트에 대한 개선을 제공하기 위해 커뮤니티 내에서의 협력을 수행하고 있습니다.

실제로 SKT는 Kubernetes에 OpenStack을 배치하여 다음과 같은 많은 혜택을 보았습니다.

- 쉽고 간단한 설치.
- 클러스터 자동 회복.
- 실행 중인 서비스에 미치는 영향을 최소화하며 OpenStack을 업그레이드와 업데이트할 수 있는 기능.
- Blue-Green 배포, Canary 릴리즈를 포함한 향상된 릴리즈 방법론들을 신속하게 채택 가능합니다.
- 컨테이너 격리를 통해 Python 종속성을 완벽하게 자동 관리.
- 비밀과 구성을 안전하게 관리.
- 클러스터 업데이트의 신속하고 유연한 출시.

SKT는 여전히 이 접근법을 테스트하고 있지만, production에 OpenStack-Helm 배포를 적극적으로 추진하고 있습니다. 올해 말까지 SKT는 최소한 3개의 production cluster를 갖게되며, 네 번째로 큰 규모의 온라인으로 2019년부터 제공됩니다. 이러한 사용 사례에는 다음이 포함됩니다.

- 빅 데이터 플랫폼 (2018년 4분기에 라이브에 출시 예정)
- 가상 데스크탑 인프라 플랫폼 (2018년 4분기에 프로덕션 준비)
- 범용 내부 프라이빗 클라우드 (2018년 3분기에 프로덕션 준비)
- 가상 네트워크 기능을 기반으로 구축된 통신 네트워크 인프라 (2019년 언젠가 작업할 예정)

SKT is also trying to improve automation on telecom infrastructure operation by utilizing containerized VNFs and leveraging containers' auto healing and fast scale-out features. In order to allow interaction between virtual machine based VNFs and containerized VNFs, Simplified Overlay Network Architecture(SONA), which is a virtual network solution for OpenStack, will support communication between VMs and containers. SONA uses the Kuryr project for integration of OpenStack and Kubernetes, and it optimizes network performance using software defined networking technologies.

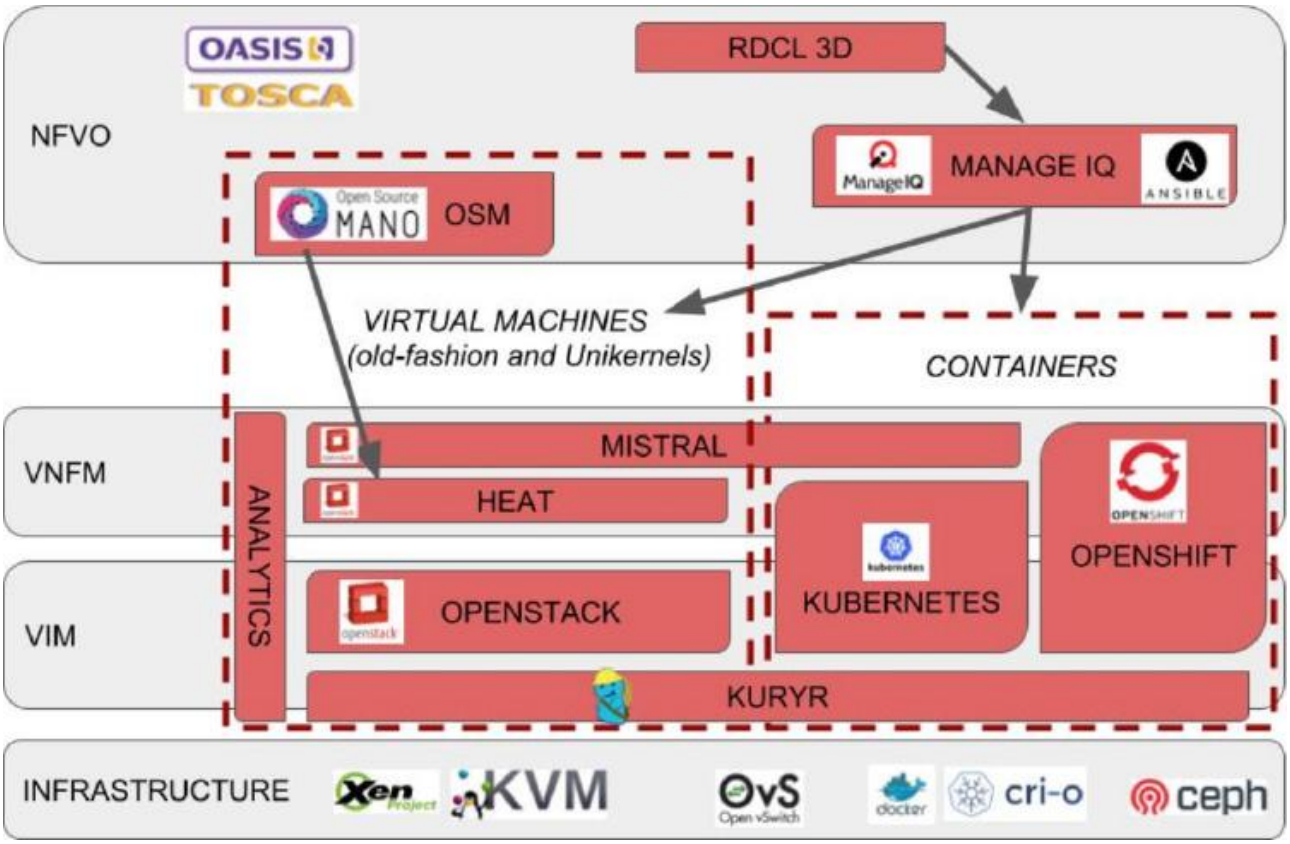
전반적으로, SKT는 Kubernetes가 OpenStack을 배포하고 운영하는 많은 복잡한 문제를 해결하는 데 도움이 된다는 사실을 알고 있습니다. OpenStack을 단순화하면서 5G시대를 위한 고급 인프라 혁신을 제공하는 강력한 접근 방식을 제공합니다. Kubernetes의 Openstack에 중점을 두는 노력은 컨테이너의 마이크로서비스로의 진화하는 전환을 처리하고 인공지능, 사물의 인터넷 및 기계학습을 제공하는 데 중요한 인프라가 되는 내부 기능을 크게 향상시켰습니다.

Superfluidity

Superfluidity 프로젝트는 유럽 12개국 18개 파트너로 구성됩니다. 즉석에서 서비스를 인스턴스화하고 네트워크(core, aggregation, edge)의 어느 곳에서나 실행하고 다른 위치로 투명하게 이동하는 기능을 향상시키는 것을 목표로 합니다. SUPERFLUIDITY는 5G 네트워크를 위한 기본 인프라 블록을 구축하려는 유럽 연구 프로젝트(Horizon 2020)입니다. 잘 알려진 오픈 소스 프로젝트를 활용하고 확장합니다. SUPERFLUIDITY는 모바일 엣지에서 혁신적인 유스케이스를 가능하게하고, 새로운 비즈니스 모델에 힘을 부여하며, 투자 및 운영 비용을 절감할 수있는 통합된 클라우드 기반 5G concept을 제공합니다.

이러한 목표를 달성하기 위해 프로젝트 컨소시엄은 기존의 VM 기반 응용 프로그램에서 Cloud Native 컨테이너 응용 프로그램으로 전환하고 있습니다. Kuryr은 OpenStack 가상 시스템과 Kubernetes 및 OpenShift 컨테이너 서비스 간의 다리 역할을 합니다.

이 프로젝트는 중앙 네트워크 기능 가상화 오케스트레이션(NFVO), 응용 프로그램 배포 및 수명주기 관리, OpenStack 서비스로서의 ManageIQ를 사용합니다. VM, 컨테이너 통합을 위한 OpenShift를 통해 Heat, Neutron 및 Octavia 및 Kubernetes를 비롯한 여러 가지 주요 기능을 지원합니다.



ManageIQ 어플라이언스에서 실행되는 Ansible 플레이 북을 활용하여 SUPERFLUIDITY는 애플리케이션을 배포하는 일반적인 방법을 제공합니다. 이러한 응용 프로그램은 OpenStack Heat 템플릿과 OpenShift 템플릿이 제공하는 클라우드 오케스트레이션 기능을 사용합니다.

그 컨소시엄은 컨테이너 내에 5G 클라우드 무선 액세스 네트워크(CRAN) 및 모바일 엣지 컴퓨팅(MEC) 구성 요소를 배포합니다. 또한 분산 인프라 스트럭처 위에 비디오 스트리밍과 같은 고처리량(high throughput) 애플리케이션을 배포합니다.

클라우드 고유의 애플리케이션 전달 방식으로 전환하면 신속하고 탄력적 인 수퍼 플로리티(SUPERFLUIDITY) 설치가 가능합니다. 이를 통해 VM 기반 응용 프로그램과 구성 요소를 컨테이너로 원활하게 전환하면서 특정 응용 프로그램에서 VM을 사용할 수있는 융통성을 유지할 수 있습니다. 이러한 응용 프로그램의 예로는 단일 경로 입/출력 가상화(SRIOV)에 필요한 특수 보안 보호 또는 네트워크 가속이 있습니다.

규모의 성능 테스트에서 SUPERFLUIDITY는 초당 22 포드의 속도로 약 1000 포드를 발사 할 수있었습니다(생성에서 실행까지의 시간 측정). 이 놀라운 성능은 OpenStack이 관리하는 VM에서 OpenShift를 실행하여 Kuryr이 이중 캡슐화 성능 저하를 피하기 위해 포드 네트워크 드라이버 역할을 수행함으로써 달성되었습니다.

Select Another Case Study

IV. 결론

지난 몇 년 동안, 컨테이너가 개발자와 조직 모두에게 중요한 도구가 됨에 따라 OpenStack은 모듈 디자인과 광범위한 커뮤니티를 활용하여 다양한 수준의 컨테이너 기술을 통합했습니다. 컨테이너와 OpenStack을 프로덕션으로 가져 오는 다양한 조직과 컨테이너와 함께 새로운 기능을 제공하는 프로젝트의 수를 볼 수 있습니다. OpenStack Foundation은 OpenStack 내에서 신기술을 통합하고 활용할 수 있도록하기 위해 노력하고 있으며, 컨테이너는 그 약속의 중요한 예입니다.

더 자세한 내용을 위해서는 컨테이너 방문 페이지를 방문하십시오.이 페이지에서 이 문서의 사본과 수십 개의 동영상 링크를 찾을 수 있습니다. OpenStack과 컨테이너의 통합에 중점을 둡니다. Kubernetes와 OpenStack 통합을 구축하고있는 커뮤니티와 직접 참여하는 경우를 위해, Kubernetes SIG-OpenStack에는 슬랙 채널, 메일링 리스트 및 주간 회의가 활용됩니다.

V. 오픈소스 프로젝트 인덱스

Airship

Airship은 상호 운용 방식으로 느슨하게 결합된 오픈 소스 도구 모음이며, 그러한 도구들은 응용 프로그램 플랫폼 형태의 Kubernetes를 기반으로, 선언적 방식으로 자동화 된 클라우드 프로비저닝 기능과 관리 기능을 제공합니다.

Ansible

Ansible은 OpenStack을 설치하고 관리 및 배포하는 데에 일반적으로 사용되는 오케스트레이션 도구입니다.

Cinder

OpenStack Cinder는 70 가지가 넘는 다양한 스토리지 드라이버를 지원하는 단일 API를 제공하는 서비스 형태의 블록 스토리지를 제공합니다.

OpenStack 클라우드 제공자

Cloud Provider OpenStack은 Kubernetes Cloud Provider 인터페이스의 구현입니다. Cloud Provider OpenStack은 OpenStack에서 호스팅하는 Kubernetes 클러스터가 OpenStack 클라우드의 스토리지와 로드 밸런서 리소스에 직접 액세스 할 수 있도록 해줍니다.

Calico

Calico는 Kubernetes 및 OpenStack를 위한 L3 전용 라우팅 기능이있는 네트워크 오버레이입니다.

Cyborg

Cyborg는 FPGA, GPU, ASIC 및 기타 하드웨어 가속기에 대한 일반적인 관리 프레임워크를 제공하는 OpenStack 프로젝트입니다. 포드(pod)에 대한 일반적인 하드웨어 인터페이스를 나타내기위한 작업이 현재 진행 중입니다.

Docker

Docker는 컨테이너화된 애플리케이션을 호스팅하는 데에 사용되는 오픈소스 컨테이너 가상화 프레임워크입니다.

Helm

Helm은 Kubernetes의 공식 패키지 관리자입니다. 응용 프로그램 배포는 Kubernetes 클러스터에서 자동으로 배포 및 관리될 수 있는 Helm-Charts에 의해 진행됩니다.

Ironic

Ironic은 OpenStack 베어메탈 서비스입니다. 독립 실행형 서비스 형태 또는 OpenStack Nova의 드라이버 형태로 실행되는이 제품은 등록, 프로비저닝, 유지보수 및 폐기를 포함하여 베어메탈 시스템의 전체 라이프 사이클을 관리할 수 있습니다.

Loci

LOCI는 OpenStack 프로젝트를 위한 경량의 OCI 호환 컨테이너를 만들기 위한 OpenStack 프로젝트입니다.

LXC

LXC는 Linux 커널 네임스페이스 격리(isolation) 및 다른 기술들을 활용하여 isolated Linux 런타임을 만드는 저수준 컨테이너 가상화 인터페이스입니다.

Kata Containers

Kata Containers는 컨테이너처럼 느껴지고 수행되지만 가상머신(VM)의 작업 부하 격리 및 보안 이점을 제공하는 경량 VM의 표준 구현입니다.

Keystone

Keystone은 OpenStack 클라우드 환경에 대한 사용자 계정 및 역할 정보를 인증하고 관리하는 수단을 제공하며 Kubernetes을 비롯한 다른 환경에 대한 플러그인으로도 사용될 수 있는 OpenStack Identity 서비스입니다.

Kolla (Containers)

Kolla(Containers)는 각 OpenStack 서비스에 대한 컨테이너를 작성하는 OpenStack 프로젝트입니다. 정교한 빌드 및 템플릿 시스템을 포함하며 다양한 호스트 운영 체제에서 소스 및 패키지 모두로부터 컨테이너를 작성할 수 있습니다.

Kolla Ansible

Kolla Ansible은 Kolla 컨테이너를 통해 OpenStack을 전체 설치하고 유지 관리하기 위해 Ansible을 사용하는 OpenStack 프로젝트입니다.

Kubernetes

Kubernetes는 클라우드 인프라를 기반으로 robust하고 고가용성의 애플리케이션을 제공하는 컨테이너 오케스트레이션 시스템입니다.

Kuryr

Kuryr은 Docker 및 Kubernetes를 포함한 컨테이너 런타임에 Neutron 네트워크 오버레이를 제공하는 OpenStack 프로젝트입니다. 컨테이너 및 VM 네트워크를 위한 "통합 브릿지"를 목표로 합니다.

Magnum

Magnum은 Kubernetes, Docker Swarm, Mesos 및 DC/OS 플랫폼을 포함한 관리 컨테이너 플랫폼을 서비스로 제공하는 OpenStack 프로젝트입니다. 간단한 사용자 지향 API를 통해 테넌트가 분리된 응용 프로그램 플랫폼을 만들 수 있습니다.

Neutron

Neutron은 OpenStack software-defined networking service로 수십 개의 네트워크 드라이버에 의해 지원되는 동적 네트워크 인프라를 제공하는 단일 API를 제공합니다.

OpenStack Ansible

OpenStack Ansible은 LXC 컨테이너에 OpenStack 서비스를 구축하고 이러한 컨테이너 서비스 내에 OpenStack 설치를 배포 및 관리하는 프로젝트입니다.

OpenStack Helm

OpenStack Helm은 OpenStack의 라이프 사이클과 Kubernetes(예 : Ceph 및 MariaDB)의 인프라 지원 인프라를 배포하고 관리하는 OpenStack 프로젝트로 소규모 배포에서부터 대규모 중앙 사무소에 이르기까지 광범위한 사용 사례에 대비 한 생산 준비 배포를 제공합니다. Helm 패키지 관리 시스템 활용하여. OpenStack Helm은 베어 메탈(Ironic) 및 가상(Nova / KVM) 작업 부하 관리를 모두 지원하며 LOCI 및 Kolla 컨테이너를 지원하는 image agnostic입니다.

Qinling

Qinling은 Functions as a Service를 제공하는 OpenStack 프로젝트입니다. Qinling은 Kubernetes 및 Docker Swarm과 같은 다양한 컨테이너 오케스트레이션 플랫폼은 물론 로컬 파일 저장소, OpenStack Swift 및 S3와 같은 다양한 function package storage backend를 지원합니다.

Triple-O

TripleO는 OpenStack의 클라우드 서비스를 기반으로 OpenStack 클라우드를 설치, 업그레이드 및 운영하기 위한 프로젝트입니다. Nova, Ironic, Neutron, Heat 및 Cloud Management를 자동화 할 수있는 Ansible을 기반으로 합니다.

Zun

Zun은 OpenStack Containers 서비스입니다. 서버 또는 클러스터를 관리할 필요없이 응용 프로그램 컨테이너를 실행하기 위한 API 서비스를 제공하는 것을 목표로 합니다.

VI. 저자

OpenStack SIG-Kubernetes 커뮤니티 회원

- Jaesuk Ahn, SK Telecom
- Christian Berendt, Betacloud Solutions GmbH
- Anne Bertucio, OpenStack Foundation
- Pete Birley, AT&T
- Chris Hoge, OpenStack Foundation
- Lingxian Kong, Catalyst Cloud
- Hongbin Lu, Huawei
- Daniel Mellado, Red Hat, Inc.
- Allison Price, OpenStack Foundation
- David Rabel, B1 Systems GmbH
- Sangho Shin, SK Telecom
- Davanum Srinivas, Huawei
- Luis Tomás, Red Hat, Inc.
- Sam Yaple, Verizon Digital Media Services
- Mikhail Fedosin, Red Hat, Inc.
- Flavio Percoco, Red Hat, Inc.

편집자

- Brian E Whitaker, Zettabyte Content LLC

Translators

- Ian Y. Choi, <ianyrchoi@gmail.com>
- Jaeho Cho, <jaeho8032@gmail.com>
- Jaewook Oh, <kyle.oh95@gmail.com>

- Soonyeul Park, <ardentpark@gmail.com>
- Sungjin Kang, <ujuc@ujuc.me>
- Sungwook Choi, <csucom@gmail.com>