One Year with DC/OS

민영근, 공학박사
AJ네트웍스 IT센터 아키텍처팀

목차

- 소개 및 도입 목적
- 구축
  - H/W, S/W
- 운영
  - 응용 프로그램 배포하기
  - 기존 시스템과 연동하기
  - 업그레이드 하기
- 서비스 중인 시스템 소개
  - 홈페이지, 그룹웨어, 전자계약
- 결론

민영근

- AJ 네트웍스 지주부문 IT센터 아키텍처 팀
- 2014.8 ~ 2016.6 kt NexR 개발팀
- 2013.8 ~ 2014.8 kt NexR TA팀
- 2011.8 ~ 2013.7 단국대학교 연구전담 교수
- 2011.2 단국대학교 공학박사

https://github.com/minyk

Drake Youngkun Min
minyk

Add a bio

AJ Networks
Seoul, Korea
https://www.linkedin.com/in/y...

Overview | Repositories 113 | Stars 997 | Followers 35 | Following 38

Pinned repositories — Customize your pinned repositories

- **nifi-sandbox** — Sandbox for Apache nifi — Shell ★12 ⑂4
- **mesosphere/universe** — The Mesosphere Universe package repository. — HTML ★282 ⑂416
- **dcos/examples** — DC/OS examples — Shell ★118 ⑂138
- **dcos-etcd** — Etcd Scheduler for DC/OS — Java ★1
- **dcos-redis** — Forked from mesosphere/dcos-commons — Simplifying stateful services — Java
- **spark-notebook-sandbox** — Sadnbox of Spark-notebook — Shell ★10 ⑂4

AJ가족

- 1960년 설립한 아주산업(주)을 전신
- 2007년 아주그룹에서 계열 분리
- 현재 AJ네트웍스, AJ셀카, AJ토탈, AJ파크 외 10개사 총 매출 1.5조원
- B2B, B2C 전 영역에 걸쳐서 렌탈 사업 진행
  - 자동차 / 건설장비
  - 고소장비 / 파렛트
  - 냉장 창고
  - 주차장
- 신규 스타트업 투자/ 인수
  - AJ전시몰
  - 링커블

AJ가족

도입 목적

- 반복적인 일
  - 고객의 반응
  - 새로운 문제의 해결
  - 새로운 기술에 집중
- 매번 반복되는 일
  - 미리 결정
  - 기반을 형성
- 미리 문제 될 만한 것
  - 사전에 차단
  - 번거로운 일의 최소화

구축

Brief History at AJ가족

- 2016.08 컨테이너 오케스트레이션 서베이
- 2016.09 DC/OS 기능 테스트 시작 (VM)
- 2017.04 DC/OS 파일럿 클러스터 시작 (5 노드)
- 2017.10 DC/OS 운영 클러스터 도입 (29 노드)
- 2018.03 DC/OS 기반의 서비스 운영 개시

구축

- 하드웨어 도입
  - 2017년 9월 29일
  - OS설치 및 초기 테스트 진행

- DC/OS 설치
  - 2017년 10월 18일
  - DC/OS 버전: 1.10.0

구축: H/W

- 네트워크
  - 관리망: 1G, 원격 서버 관리용
  - 서비스망: 1G * 2, Active-backup
  - 클라우드 망: 10G * 4, Teaming, 폐쇄망

- 서버
  - 관리/ 마스터/ 에이전트
  - 10 Core * 2 ea, 64/ 128/ 256 GB, 4 TB * 10 ea

구축: S/W

- DC/OS 설치
  - Overlay 네트워크 생성: 17개
  - Rexray: Ceph 설정
- DC/OS 설치 후 작업
  - Agent 노드의 CPU/ Memory 조절: 2 Core/ 10 GB 제외
  - CPU 제한 완화: Hard limit -> Soft limit
  - Marathon-on-Marathon 구성
- 모니터링
  - 메트릭: dcos-metrics, cadvisor + Prometheus + Grafana
  - 로그: logstash + elasticsearch + kibana(with LogTrail)
- 저장소
  - Ceph(Agent 노드와 colocate)

컨테이너 기반의 사내 클라우드 서비스

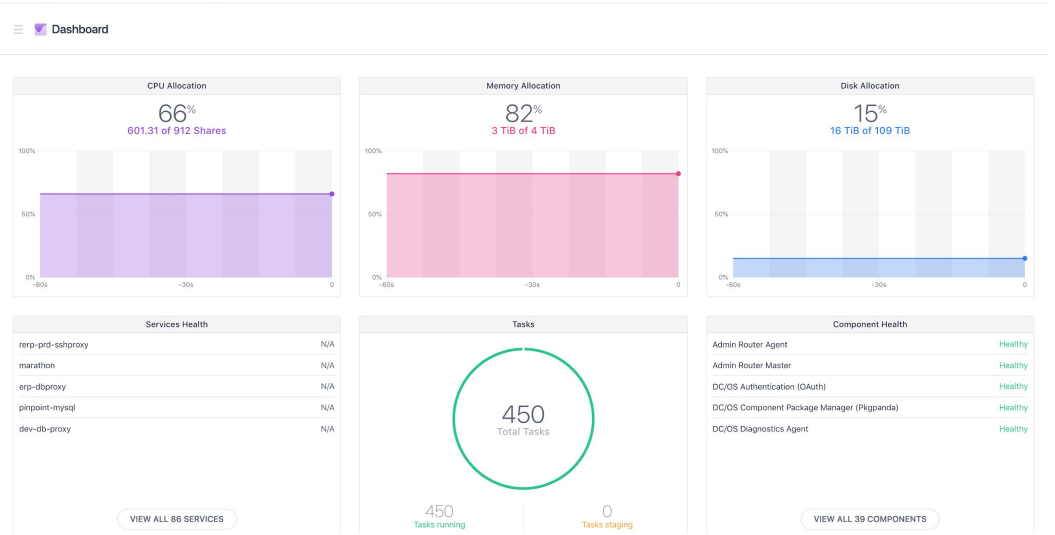
- 핵심 구성 요소
  - DC/OS
  - Ceph
  - Monitoring/ Alerting
  - HTTP Proxy/ DMZ
  - Harbor, Gitlab
- 고객
  - 사내 서비스 운영팀
- 구축 및 운영
  - IT센터 아키텍처팀 6명

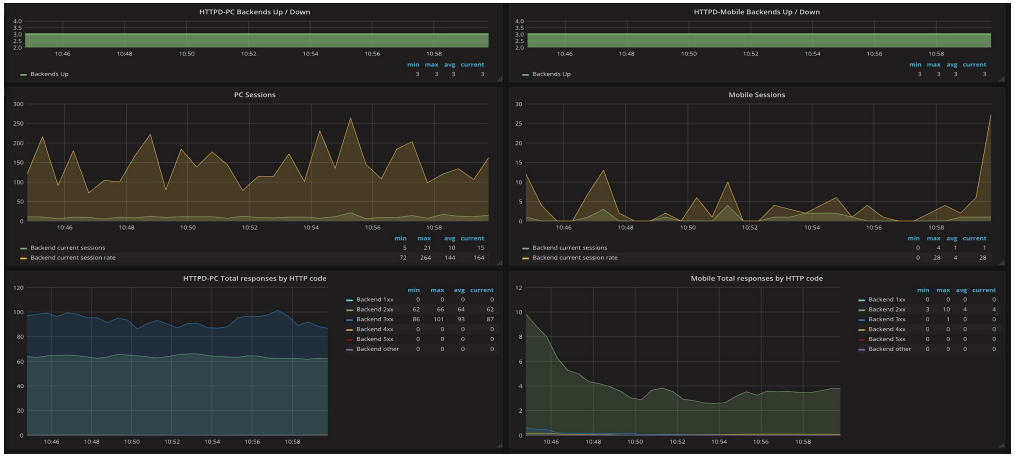
운영

운영: 모니터링

- 매일
  - 태스크 상태
- 매주
  - 네트워크, 노드 상태

- 알람 연동
  - 각 노드의 상태
  - 태스크 상태
  - Marathon-LB의 HAProxy 메트릭





운영: 서비스 배포하기

- 서비스 개발 초기에 투입하여 개발/ 운영 환경 설정
  - 개발물의 컨테이너화(== Dockerfile)
  - 배포용 JSON 템플릿 개발(== deploy.json)

- CI/CD를 사용하여 자동/반자동 배포 설정
  - CI/CD 설정(== .gitlab-ci.yaml)
  - 컨테이너화한 후 Marathon API로 자동 배포
  - 배포 후 작업 자동화를 위한 프로그램 개발/ 적용

운영: 다른 시스템과 연동하기

- 신규 서비스만 DC/OS에서 운영, 기존 시스템들과 연동 필수
  - Incoming
    - HTTP/HTTPS 통신: 도메인을 사용하도록 권고
    - TCP 통신: SOCKS Proxy 사용
  - Outgoing
    - HTTP 통신은 도메인을 사용 후, 관리노드에 구축한 HTTP Proxy 사용
    - TCP 통신은 해당 IP/Port 로 연결하는 Proxy를 public 노드에 별도 배포

- DC/OS는 모든(대부분) 것이 도메인 이름 기반
  - xxx.mesos
  - xxx.autoip.dcos.thisdcos.directory
  - xxx.l4lb.thisdcos.directory

운영: 업그레이드 하기

- 최초 설치 이후 현재까지 업그레이드 2회 수행
  - 2018.2: 1.10.4
  - 2018.8: 1.10.8

- 무중단 업그레이드 가능
  - 실 서비스에 대한 영향없이 업그레이드 하려면 조금 더 고민 필요
  - 개별 노드의 업그레이드 진행을 확인 어려움

서비스 중인 시스템

서비스 형식

- SaaS 형식
  - DB, Etcd, ActiveMQ 등
  - 아키텍처팀에서 관리, 서비스 형태로 제공
- PaaS 형식
  - 컨테이너 템플릿, 배포 설정은 아키텍처팀에서 제공
  - 운영팀은 배포 권한, 서비스 모니터링 수행
- IaaS 형식
  - Marathon UI 제공
  - 컨테이너화, 배포 설정을 운영팀에서 직접 관리

서비스중인 시스템

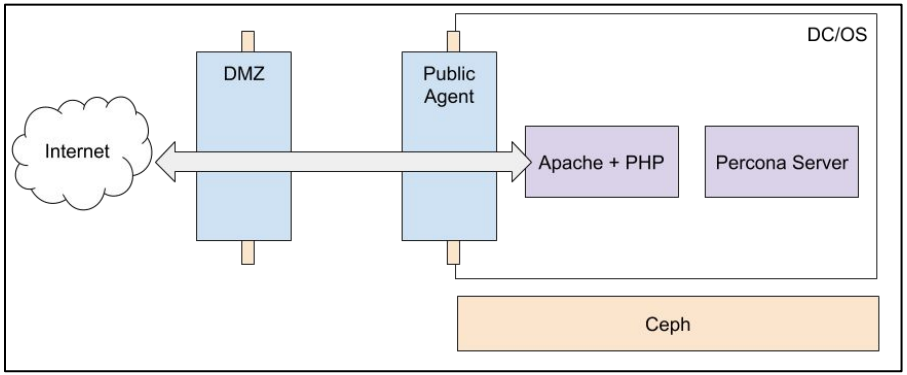
- 홈페이지
- 그룹웨어
- 전자계약

서비스 중인 시스템: 홈페이지

- 요건
  - 전형적인 웹 홈페이지: Apache, PHP, MySQL
  - 인터넷 망에서 접속 필요
  - 유지보수 업체 별도로 있음
  - 홈페이지 개/보수하면서 DC/OS로 이전

서비스 중인 시스템: 홈페이지

- 작업
  - Apache + PHP, Percona Server 컨테이너 배포
  - Gitlab CI/CD
    - 개발 시스템 자동 배포
    - 운영 시스템 반자동 배포
  - DMZ 서버에 TLS 인증서 적용
  - 운영 담당자/ 유지보수 업체 담당자 교육
- 배포 컨테이너
  - 2개, 총 2 Core, 12 GB 메모리
- 주요 이슈
  - 로컬 스토리지를 인스턴스들이 공유 필요: CephFS 적용

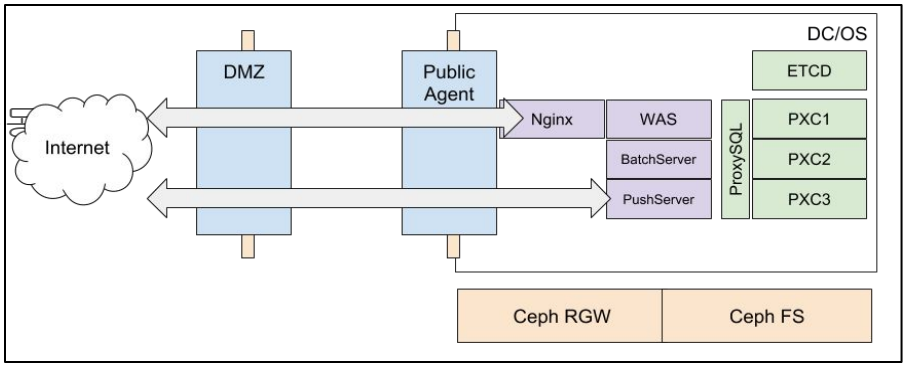


서비스 중인 시스템: 그룹웨어

- 요건
  - 상용 제품 도입
  - 3 티어 구조: nginx, tomcat, db
  - Mobile Push 서버
  - Web/WAS 다중화
  - Batch 처리 있음
  - 메일 서버/ 검색 엔진

서비스 중인 시스템: 그룹웨어

- 작업
  - 제품의 컨테이너 이미지 제작
  - Batch 기능 별도 분리: 1개 인스턴스로 구동
  - 메일 서버와 검색 엔진은 dedi로 구동
  - DB: Percona XtraDB Cluster + ProxySQL + ETCD 사용
  - Gitlab CI/CD 구성
- 배포 컨테이너
  - 21개, 총 86 Core, 717 GB 메모리
- 주요 이슈
  - marathon-lb의 설정 로딩 문제: marathon-lb v1.12 에서 해결
  - L4LB 도메인의 연결 끊김 문제: 각 버전별 최신 릴리즈에서 해결

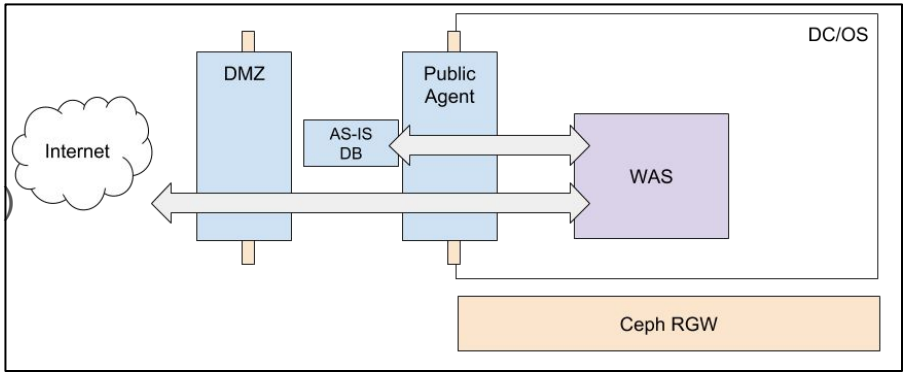


서비스 중인 시스템: 전자계약

- 요건
  - 자체 개발, WAS
  - 서비스 상태 제어 (운영팀)
  - 기존 시스템 연동

서비스 중인 시스템: 전자계약

- 작업
  - 별도의 Marathon 구동/ LDAP 플러그인 적용
  - 개발자 교육(오버레이 네트워크, 배포 등)
  - 기존 시스템과 통신할 수 있도록 구성
  - Gitlab CI/CD: 운영팀에서 직접 구성
- 배포 컨테이너
  - 1개, 총 1 Core 4 GB 메모리
- 주요 이슈
  - 표준화된 Docker Base 이미지(Tomcat 등) 제공



서비스 운영 담당자 한 마디

- 그룹웨어 서비스 운영 담당자
  - 서비스 무중단 배포: 프로그램 배포 시 서비스 무중단으로 배포가 가능하여 야근시간 단축
  - 빠르고 손쉬운 복구: 배포한 프로그램을 이전으로 복구 시 UI환경에서 빠르고 쉽게 이전 버전으로 복구가능

- 전자계약서 서비스 운영 팀장
  - 개발/ 운영 환경의 (강제적인) 표준화
  - 신규 서비스 개발 시 서버 준비과정이 줄어들어 업무 효율 증가
  - 개발/운영 업무의 자동화,표준화 의지가 없다면 쓸데없는 업무 추가
  - 충분한 가이드가 없을 경우, 오히려 개발환경 구축이 더 복잡

결론

구축/ 운영 성과

- 실 서비스 운영
  - 그룹웨어 외 8개 서비스

- DC/OS 생태계 기여
  - DC/OS SDK를 사용한 서비스 자체 개발: dcos-etcd, dcos-redis
  - mesosphere/universe, dcos/examples
  - DC/OS JIRA 이슈 등록
  - Google Groups 활동

향후 계획

- 렌터카 ERP 런칭
- DC/OS 업그레이드: 1.11 또는 1.12
- 재난복구 구성: 멀티 클러스터
- 서비스 카달로그 제공

결론

- 좋은 점
  - 서비스 운영과 인프라 운영 확실히 분리
  - 가장 깔끔한 인스톨러 및 업그레이드 방법

- 어려운 점
  - 컨테이너 및 DC/OS 기반에 대한 교육 비용이 높음
  - DC/OS 운영에는 다양한 분야의 경험 필요

Any Questions?