Knowledge Utility Kit for Global Intelligence Integration

Gini

2025년 11월 17일

☒ Download PDF

CHAPTER 1

Agent Payment Protocol Samples

1.1 Introduction

1.1.1 AP2 프로토콜 개요

AP2(Agent to Payment)는 Google이 제안한 AI 에이전트 시대의 결제 프로토콜입니다. AI 에이전트가 사용자를 대신하여 자율적으로 상품을 구매하고 서비스에 가입하는 미래를 대비하여, 안전하고 효율적인 결제 방법을 정의합니다.

1.1.2 세 가지 시나리오

AP2 샘플은 서로 다른 특성을 가진 다음 시나리오들을 제공합니다.

- Cards 시나리오 (페이지 **??**) - 전통적인 신용카드 결제 네트워크를 통한 AI 에이전트 자동 결제
- DPC 시나리오 (페이지 **??**) - 디지털 자격증명 기반 차세대 결제 방식
- x402 시나리오 (페이지 **??**) - HTTP 402 상태 코드를 활용한 웹 네이티브 결제

1.1.3 Cards: 기존 인프라 활용

Cards 시나리오는 기존의 **카드 결제 네트워크**(Visa, Mastercard 등)를 통해 AI 에이전트가 자동으로 결제를 수행하는 방식입니다. AP2의 세 가지 시나리오 중 **가장 현실적이고 즉시 도입 가능한** 접근 방식으로, **이미 검증된 기존 카드 결제 인프라**를 그대로 활용하여 빠른 도입과 높은 호환성을 제공합니다.

주요 특징

1. **기존 인프라 활용**: Visa, Mastercard, AMEX 등 전 세계적으로 구축된 카드 네트워크 사용
2. **즉시 적용 가능**: 새로운 표준이나 프로토콜 도입 없이 기존 시스템과 호환
3. **높은 신뢰성**: 수십 년간 검증된 보안 및 사기 방지 시스템
4. **PCI DSS 준수**: 카드 산업 데이터 보안 표준을 따르는 안전한 처리
5. **광범위한 수용성**: 대부분의 가맹점에서 이미 지원하는 결제 수단

사용 사례

- **구독 서비스 자동 결제**: AI 에이전트가 필요한 서비스를 자동으로 구독하고 결제
- **일상 구매**: 온라인 쇼핑에서 에이전트가 가격 비교 후 자동 구매
- **B2B 거래**: 기업용 카드를 통한 자동 발주 및 결제
- **동적 예산 관리**: 설정된 예산 범위 내에서 최적의 구매 결정

한계점

- **수수료**: 카드 네트워크 수수료 (보통 2-3%) 존재
- **보안 요구사항**: PCI DSS 준수를 위한 추가 보안 조치 필요
- **지역별 제약**: 일부 국가에서는 카드 사용 제한 또는 높은 수수료
- **개인정보**: 카드 정보 저장 및 관리에 대한 규제 준수 필요

1.1.4 DPC: 디지털 자격증명 기반 결제

DPC(Digital Payment Credential) 시나리오는 **암호학적으로 검증 가능한 디지털 자격증명**을 결제 수단으로 활용하는 혁신적인 접근 방식입니다. Cards가 기존 카드 네트워크에 기반한다면, DPC는 **EUDI Wallet과 같은 디지털 신원 표준**을 결제 영역으로 확장한 것입니다.

샘플 시나리오는 Android 앱이므로 Android Credential Manager API를 통해 상호작용하며, 사용자는 암호학적으로 서명된 증명(cryptographic proof)을 제시하여 결제를 완료합니다. 이는 단순히 카드 정보를 전달하는 것이 아니라, **사용자가 결제카드 소유자이며 이 거래을 승인했음을 증명**하는 방식입니다.

주요 특징

1. **암호학적 보안**: 공개키 암호화 기반의 서명으로 위변조 불가능
2. **선택적 공개 (Selective Disclosure)**: 필요한 정보만 선택적으로 공개
3. **사용자 통제**: 모든 거래에서 사용자가 명시적으로 승인
4. **표준 기반**: OpenID4VP, ISO 18013-5 mDOC, DCQL 등 국제 표준 준수
5. **프라이버시 보호**: 최소한의 정보만 공유하며, 추적 방지 메커니즘 내장

사용 사례

- **프라이버시 중심 결제**: 최소한의 개인정보만 공개하면서 결제
- **규제 준수 환경**: GDPR, EUDI Wallet 등 엄격한 프라이버시 규제가 있는 EU 시장
- **고가 거래**: 강력한 인증이 필요한 고액 결제
- **신원 연계 결제**: 결제와 동시에 연령 확인이나 자격 증명이 필요한 경우
- **크로스보더 거래**: 국제 표준 기반으로 국경을 넘는 거래에 적합

한계점

- **플랫폼 한정**: 현재 Android 에만 구현, iOS 지원 제한적
- **생태계 미성숙**: Credential provider가 아직 초기 단계
- **사용자 교육**: 새로운 개념으로 사용자 이해와 신뢰 구축 필요

- **표준화 진행 중**: OpenID4VP, mDOC 등이 아직 완전히 확립되지 않음

1.1.5 x402: HTTP 네이티브 결제

x402는 1997년 HTTP/1.1 표준에 정의되었지만 거의 사용되지 않았던 **HTTP 402 상태 코드**를 현대적인 AI 에이전트 생태계에 맞게 재해석한 프로토콜입니다. AP2 샘플의 x402 시나리오는 이 개념을 구현한 것으로, AI 에이전트들이 웹 리소스나 서비스에 접근할 때 자동으로 결제를 처리할 수 있도록 합니다.

주요 특징

1. **AI 에이전트 친화적**: 사람의 개입 없이 에이전트가 자율적으로 결제 결정
2. **프로토콜 독립적**: 특정 결제 수단이나 통화에 종속되지 않음
3. **표준 기반**: HTTP 표준을 확장하여 웹 생태계와 자연스럽게 통합
4. **마이크로 페이먼트**: 소액 결제에 최적화 (거의 무료에 가까운 수수료)
5. **중개자 불필요**: 판매자와 구매자 간 직접 거래 조건 설정 가능

사용 사례

- **AI 에이전트 모니터링 서비스**: 에이전트의 활동 추적 및 성능 분석에 대한 결제
- **프리미엄 API 접근**: 유료 API나 데이터셋에 대한 자동 결제
- **컴퓨팅 리소스**: 필요한 만큼만 사용하고 즉시 결제하는 온디맨드 서비스
- **콘텐츠 마이크로 페이먼트**: 뉴스 기사, 연구 논문 등 개별 콘텐츠에 대한 소액 결제

한계점 및 고려사항

- **표준 부재**: HTTP 402는 1997년부터 정의되었으나 실제 구현 사례 거의 없음
- **생태계 부재**: 결제 프로세서, 가맹점, 클라이언트 모두 새로 구축 필요
- **보안 표준**: 아직 확립된 보안 표준이나 모범 사례 부족
- **채택 장벽**: 기존 시스템과의 호환성 없어 전면적인 생태계 전환 필요
- **불확실성**: 실험적 단계로 장기적 실용화 여부 불확실

x402의 미래 전망

단기 (2025-2027):

- 실험적 파일럿 프로젝트 및 PoC (Proof of Concept)
- 특정 도메인(예: AI 서비스 마켓플레이스)에서 제한적 사용
- 표준화 논의 시작 (IETF, W3C)

중기 (2027-2030):

- 마이크로서비스 및 API 경제에서 점진적 채택
- 암호화폐 및 블록체인 기반 결제 시스템과의 통합
- 주요 클라우드 제공업체의 실험적 지원

장기 (2030+):

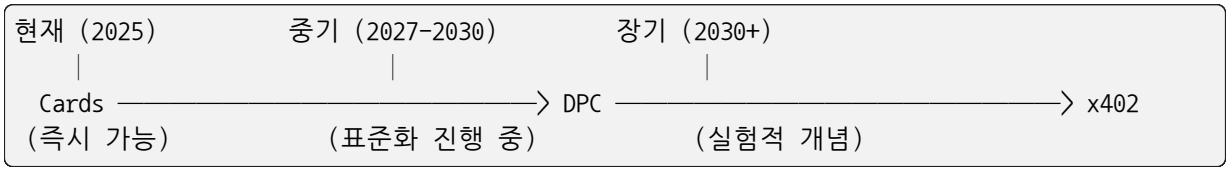
- AI 에이전트 생태계의 표준 결제 프로토콜로 자리잡을 가능성
- 웹의 근본적인 비즈니스 모델 변화 (광고 기반 → 마이크로 페이먼트)
- 사람 간 거래보다는 M2M 거래의 주류 프로토콜로 발전

1.2 결론

1.2.1 시나리오 비교

AP2 프로토콜은 AI 에이전트 시대의 다양한 결제 요구사항을 충족하기 위해 세 가지 차별화된 접근 방식을 제시합니다.

도입 타임라인



비즈니스 관점

특징	Cards	DPC	x402
기반 기술	카드 네트워크 (Visa, Mastercard)	Digital Credentials (ISO mDOC)	HTTP 402 + Blockchain
도입 시기	즉시 가능 (2025)	중기 (2027-2030)	장기 (2030+)
도입 난이도	낮음 (기존 인프라)	중간 (새로운 API)	높음 (신규 표준)
적용 범위	전통적 커머스	모바일 중심	웹 리소스/API
거래 비용	2-3% 수수료	매우 낮음 (네트워크 비용만)	거의 없음 (가스비만)
국제 표준	확립됨 (PCI DSS)	표준화 진행 중 (EUDI, OpenID4VP)	제안 단계 (실험적)
미래 전망	안정적 유지	성장 가능성 높음	실험적, 불확실

기술 구현 관점

특징	Cards	DPC	x402
결제 수단	토큰화된 카드 (DPAN)	Digital Payment Credential	디지털 화폐 (USDC 등)
자격 증명 저장	Credentials Provider	CM Wallet (Credential Manager)	Crypto Wallet
통신 프로토콜	AP2 via A2A	OpenID4VP via Credential Manager API	HTTP 402 + Blockchain RPC
Mandate 구조	Intent → Cart → Payment	Cart only	Cart + Blockchain Transaction
관련 에이전트	4 개 (Shopping, Merchant, Credentials, Payment)	2개 (Shopping, Merchant) + Wallet	3개 (Shopping, Merchant, Credentials) + Blockchain
플랫폼 지원	범용 (웹/모바일)	모바일	범용 (웹 우선)

보안 및 사용자 경험

특징	Cards	DPC	x402
보안 수준	PCI DSS 표준	암호학적 증명 (ES256 서명)	토큰 기반 + 블록체인 검증
프라이버시	낮음 (카드사 추적 가능)	매우 높음 (Selective Disclosure)	중간 (퍼블릭 블록체인)
사용자 인증	PaymentMandate 해시 서명 + OTP	transaction_data 서명 (Trusted UI)	트랜잭션 서명 (Wallet)
추가 인증	OTP 챌린지	기기 생체인증	없음 (서명으로 충분)
사용자 개입	초기 설정 후 최소화	매 거래마다 명시적 승인	거래 확인 (Wallet UI)
사용자 경험	익숙함 (기존 카드 결제)	학습 필요 (새로운 개념)	학습 필요 (암호화폐)
오프라인 지원	제한적	가능 (오프라인 서명 검증)	불가능 (네트워크 필수)

1.2.2 각 시나리오의 포지셔닝

1. Cards - 현재의 실용적 해결책

- ☒ 즉시 도입 가능
- ☒ 전 세계 인프라 구축 완료
- ☒ 검증된 보안 및 규제 준수
- ☒ 높은 수수료

- ☒ 프라이버시 제한

2. DPC - 균형잡힌 미래

- ☒ 실행 가능한 기술 (Android Credential Manager API 존재)
- ☒ 국제 표준화 진행 중 (EUDI Wallet)
- ☒ 프라이버시와 보안 최우선
- ☒ 플랫폼별 지원 차이
- ☒ 생태계 미성숙

3. x402 - 혁신적 비전

- ☒ 인터넷 네이티브 결제의 이상적 모델
- ☒ M2M 결제에 최적화
- ☒ 낮은 비용
- ☒ 표준 부재
- ☒ 생태계 미구축
- ☒ 채택 불확실성

1.2.3 권장 도입 전략

단계별 접근:

1. **현재 (2025)**: Cards 기반 구현
    - AI 에이전트에 카드 결제 통합
    - 빠른 시장 진입 및 사용자 확보
2. **중기 (2027-2030)**: DPC 준비 및 전환
    - EUDI Wallet 표준 확립 대기
    - iOS 지원 추가
    - 프라이버시 중시 시장 진출
3. **장기 (2030+)**: x402 모니터링
    - 표준화 진행 상황 추적
    - 실험적 파일럿 프로젝트
    - M2M 거래 특화 영역 탐색

1.2.4 결론

AP2 프로토콜의 세 가지 시나리오는 AI 에이전트 결제의 진화 경로를 보여줍니다:

현명한 전략은 Cards로 시작하여 즉시 가치를 제공하면서, DPC 표준 성숙을 추적하고, x402의 장기적 잠재력을 모니터링하는 것입니다. 각 시나리오는 상호 배타적이지 않으며, 사용 사례와 시장 상황에 따라 적절히 조합할 수 있습니다.

특히 DPC는 EUDI Wallet과 같은 대규모 정부 주도 이니셔티브의 지원을 받고 있어, **중기적으로 가장 유망한 선택지**로 평가됩니다. 프라이버시와 보안이 점점 더 중요해지는 디지털 시대에, DPC는 사용자 통제와 검증 가능성을 제공하는 차세대 표준이 될 것입니다.

1.3 References

- Official Documentation[2]
- GitHub Repository[3]
- EU Digital Identity Wallet ARF overview[4]

[2] https://cloud.google.com/blog/products/ai-machine-learning/announcing-agents-to-payments-ap2-protocol
[3] https://github.com/google-agentic-commerce/AP2
[4] https://en.wikipedia.org/wiki/EU_Digital_Identity_Wallet

CHAPTER 2

Cards 시나리오

This illustrates the complete flow of the Human Present Card Payment scenario using the AP2 framework with the A2A protocol.

2.1 Sequence Diagram

2.2 Key Components

2.2.1 1. IntentMandate Structure

The IntentMandate captures the user's shopping intent:

```
{
  "intent_mandate_id": "uuid",
  "natural_language_description": "I want to buy a coffee maker",
  "user_prompt_required": true,
  "merchants": ["merchant_agent"],
  "skus": ["coffee-maker-001"],
  "intent_expiry": "2025-11-18T10:00:00Z",
  "requires_refundability": false
}
```

2.2.2 2. CartMandate Structure

The CartMandate contains product and payment request details:

```
{
  "cart_mandate_id": "uuid",
  "merchant_name": "Example Merchant",
  "cart_expiry": "2025-11-18T10:00:00Z",
  "refund_period": "P30D",
  "payment_request": {
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
    "method_data": [{
      "supported_methods": ["CARD"]
    }],
    "details": {
      "id": "order-123",
      "total": {
        "label": "Total",
        "amount": {"currency": "USD", "value": "79.99"}
      },
      "displayItems": [
        {"label": "Coffee Maker", "amount": {"value": "69.99"}},
        {"label": "Shipping", "amount": {"value": "5.00"}},
        {"label": "Tax", "amount": {"value": "5.00"}}
      ]
    }
  },
  "merchant_signature": "signature_by_merchant"
}
```

2.2.3 3. PaymentMandate Structure

The PaymentMandate authorizes the payment:

```
{
  "payment_mandate_contents": {
    "payment_mandate_id": "uuid",
    "timestamp": "2025-11-17T12:00:00Z",
    "payment_details_id": "order-123",
    "payment_details_total": {
      "label": "Total",
      "amount": {"currency": "USD", "value": "79.99"}
    },
    "payment_response": {
      "request_id": "order-123",
      "method_name": "CARD",
      "details": {
        "token": {
          "value": "encrypted_token",
          "url": "http://localhost:8002/a2a/credentials_provider"
        }
      },
      "shipping_address": {
        "streetAddress": "123 Main St",
        "city": "San Francisco",
        "state": "CA",
        "zipCode": "94102"
      },
      "payer_email": "user@example.com"
    },
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
    "merchant_agent": "merchant_agent"
  },
  "user_authorization": "cart_hash_payment_hash_signature"
}
```

2.2.4 4. Payment Credential Token

The token contains DPAN card data:

```
{
  "value": "encrypted_payment_token",
  "url": "http://localhost:8002/a2a/credentials_provider"
}
```

The Credentials Provider decrypts this to reveal:

```
{
  "card_number": "4111111111111111",
  "expiry_month": "12",
  "expiry_year": "2025",
  "cvv": "123",
  "holder_name": "John Doe",
  "is_dpan": true
}
```

2.2.5 5. OTP Challenge

The OTP challenge structure:

```
{
  "challenge": {
    "type": "otp",
    "display_text": "The payment method issuer sent a verification code to the phone
↪number on file..."
  }
}
```

2.3 Protocol Flow Details

2.3.1 Mandate Lifecycle

1. **IntentMandate**: Created by Shopping Agent to express user's shopping intent
2. **CartMandate**: Created by Merchant Agent with product details and pricing
3. **PaymentMandate**: Created by Shopping Agent with selected payment method

2.3.2 Security Features

- **Merchant Signature**: CartMandate is signed by merchant to ensure authenticity
- **User Authorization**: PaymentMandate is signed by user to authorize purchase
- **Hash Binding**: User signature includes hashes of both CartMandate and PaymentMandate
- **Token-Based Credentials**: Payment details are never directly shared, only tokens
- **DPAN**: Tokenized card number instead of primary account number (PAN)
- **OTP Challenge**: Additional authentication layer for high-confidence transactions

2.3.3 Agent Responsibilities

Agent	Responsibilities
Shopping Agent	Orchestrates flow, manages user interaction, creates IntentMandate and PaymentMandate
Merchant Agent	Product catalog, creates and signs CartMandate, validates shopping_agent_id
Credentials Provider	Manages payment methods, provides DPAN tokens, decrypts credentials
Payment Processor	Processes payments, implements OTP challenge, charges card

2.3.4 Protocol Standards

This implementation follows several key standards:

1. **A2A (Agent-to-Agent)**: Communication protocol between agents
2. **AP2 (Agent Payments Protocol)**: Payment-specific extension to A2A
3. **Mandate Pattern**: Structured, signed data objects for cart and payment
4. **DPAN**: Tokenized card numbers for enhanced security
5. **JSON-RPC 2.0**: Message format for agent communication

2.3.5 Notes

- This is a demonstration implementation showing the card payment flow
- The signing simulation uses simple hash concatenation instead of real cryptographic signatures
- OTP validation uses a hardcoded value (123) for demo purposes
- Production implementations should:
  - Use real cryptographic signatures (e.g., JWT, JWS)
  - Implement proper certificate validation
  - Use secure OTP generation and validation

- Add fraud detection mechanisms
- Implement proper error handling and retry logic
- Store sensitive data securely (encrypted at rest)
- Use secure communication channels (TLS/HTTPS)

2.4 AI 에이전트 통합 방식

1. 토큰화된 카드 정보

```
// 에이전트가 안전하게 저장된 토큰화된 카드 정보 사용
val paymentToken = secureStorage.getPaymentToken()
val transaction = Transaction(
    amount = cartTotal,
    currency = "USD",
    paymentMethod = paymentToken,
    merchantId = merchantInfo.id
)
```

2. 자동 승인 로직

- 사전 설정된 지출 한도 내에서 자동 승인
- 한도 초과 시 사용자에게 확인 요청
- 이상 거래 패턴 감지 시 거래 중단

3. 멀티 카드 관리

- 여러 카드 중 최적의 카드 자동 선택 (포인트, 할인 등)
- 카드 잔액 및 한도 실시간 확인
- 거절 시 다른 카드로 자동 재시도

CHAPTER 3

DPC 시나리오

This illustrates the complete flow of the Digital Payment Credentials scenario using the AP2 framework with the A2A protocol.

3.1 Sequence Diagram

3.2 Key Components

3.2.1 1. OpenID4VP Request Structure

The DPC request follows the OpenID for Verifiable Presentations protocol:

```
{
  "protocol": "openid4vp-v1-unsigned",
  "request": {
    "response_type": "vp_token",
    "response_mode": "dc_api",
    "nonce": "<UUID>",
    "dcql_query": {
      "credentials": [{
        "id": "cred1",
        "format": "mso_mdoc",
        "meta": {
          "doctype_value": "com.emvco.payment_card"
        },
        "claims": [
          {"path": ["com.emvco.payment_card.1", "card_number"]},
          {"path": ["com.emvco.payment_card.1", "holder_name"]}
        ]
      }]
    },
    "transaction_data": ["<base64url-encoded-json>"],
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
    "client_metadata": {
      "vp_formats_supported": {
        "mso_mdoc": {
          "issuerauth_alg_values": [-7],
          "deviceauth_alg_values": [-7]
        }
      }
    }
  }
}
```

3.2.2 2. Transaction Data Structure

The `transaction_data` contains payment details that get signed:

```
{
  "type": "payment_card",
  "credentialIds": ["cred1"],
  "transactionDataHashesAlg": ["sha-256"],
  "merchantName": "Example Merchant",
  "amount": "US 25000.00",
  "additionalInfo": {
    "title": "Please confirm your purchase details...",
    "tableHeader": ["Name", "Qty", "Price", "Total"],
    "tableRows": [
      ["Tesla Model 3", "1", "25000.0", "25000.0"]
    ],
    "footer": "Your total is 25000.00"
  }
}
```

3.2.3 3. Security Features

- **Nonce**: Prevents replay attacks (should be validated by merchant)
- **Transaction Data Signing**: User signs over the exact transaction details
- **Trusted UI**: System-controlled UI prevents UI spoofing
- **Selective Disclosure**: Only requested claims are shared (DCQL)
- **Device Authentication**: ES256 signature from secure element
- **Issuer Authentication**: Credential validity from issuer

3.3 Protocol Standards

This implementation follows several key standards:

1. **OpenID4VP**: OpenID for Verifiable Presentations protocol
2. **ISO/IEC 18013-5**: Mobile driver's license (mdoc) format

3. **DCQL**: Digital Credentials Query Language for selective disclosure
4. **A2A**: Agent-to-Agent communication protocol (AP2 extension)
5. **Android Credential Manager**: Platform API for credential access

3.4 Comparison with EUDI Wallet

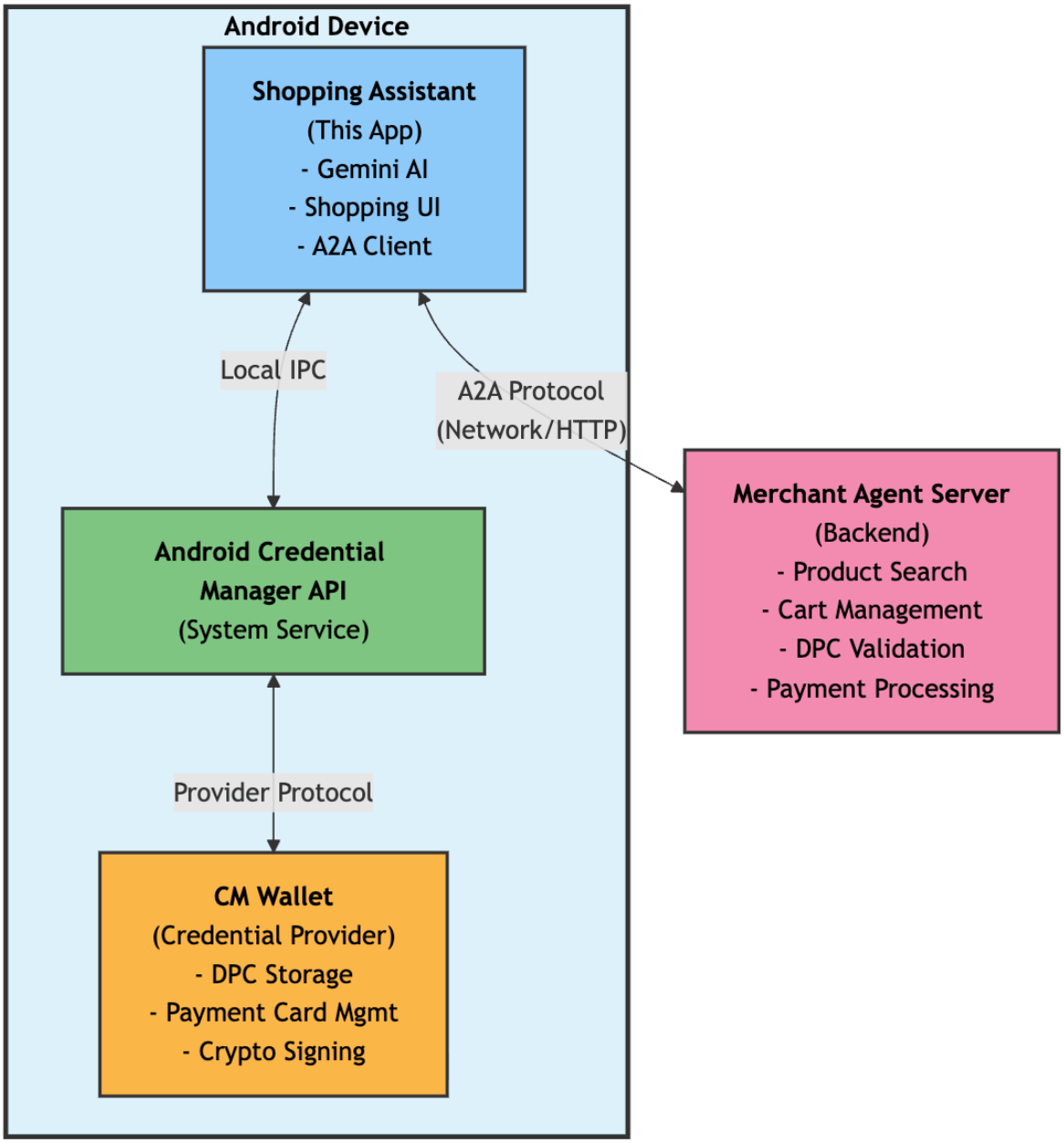
Aspect	DPC Implementation	EUDI Wallet
Protocol	OpenID4VP	OpenID4VP ✓
Format	ISO mdoc (mso_mdoc)	ISO mdoc ✓
Doctype	com.emvco.payment_card	eu.europa.ec.eudi.pid.1
Use Case	Payment authentication	Identity verification
Selective Disclosure	DCQL ✓	DCQL ✓
Signature	ES256 (COSE -7) ✓	ES256 (COSE -7) ✓
Platform	Android Credential Manager	EUDI Wallet App

3.5 Notes

- This is a demonstration implementation showing the DPC flow
- Full signature validation is marked as TODO in the merchant agent
- Production implementations should:
  - Validate all cryptographic signatures
  - Check certificate chains
  - Verify nonces and timestamps
  - Forward to payment processors for actual processing
  - Implement proper error handling
  - Add fraud detection mechanisms

3.6 CMWallet 개요

CMWallet은 Android Credential Manager 생태계에서 **Credential Provider** 역할을 하는 앱입니다. Digital Credential을 관리하는 전용 앱으로, 샘플에서는 2개의 가상 신용카드를 제공합니다.



핵심 기능:

- **Digital Payment Credentials (DPC) 보관**: 사용자의 결제 카드 정보를 디지털 자격 증명서로 보관합니다
- **Credential Manager 통합**: Android 시스템의 Credential Manager API를 통해 다른 앱에 자격 증명을 제공합니다
- **사용자 승인 UI**: 결제 승인 시 거래 세부 정보를 표시하고 사용자 확인을 요청합니다
- **암호화 서명**: 기기의 보안 요소를 사용하여 거래에 암호화 서명을 생성합니다

기술 사양:

- 사용자는 결제 시 복수의 카드 중 선택 가능
- 각 카드는 ISO 18013-5 mDOC 형식으로 저장
- ES256 알고리즘으로 서명되어 위변조 방지

왜 별도 앱인가?

- **역할 분리**: Shopping Agent는 쇼핑 경험에 집중, CMWallet은 결제 자격 증명 관리 역할에 집중합니다
- **보안 격리**: 민감한 결제 정보를 별도 앱에서 관리하여 보안을 강화합니다
- **재사용성**: 하나의 CMWallet이 여러 쇼핑 앱에서 사용 가능합니다
- **표준 준수**: Android Credential Manager의 표준 아키텍처 패턴을 준수합니다

필수 요구사항:

- CMWallet은 Shopping Agent와 동일 기기에 설치되어야 합니다

3.7 Android Credential Manager API 통합 흐름

코드 예시:

```
// Credential Manager를 통한 DPC 요청
val credentialManager = CredentialManager.create(context)
val result = credentialManager.getCredential(
    request = GetCredentialRequest(
        credentialOptions = listOf(dpcRequest)
    )
)
```

3.8 DPC 기술 상세 분석

이 예제 시나리오는 DPC (Digital Payment Credential) 요청을 생성합니다. 주요 기능은 쇼핑 카트 정보를 받아 OID4VP (OpenID 4 Verifiable Presentation) 프로토콜 기반의 인증 요청을 생성하는 것입니다.

3.9 EUID Wallet (EU Digital Identity Wallet) 표준 연관성

DPC 시나리오는 EUID Wallet 표준과 여러 측면에서 밀접하게 연관되어 있습니다

3.9.1 OID4VP (OpenID for Verifiable Presentaiton) 프로토콜 사용

예시 1: DpcHelper.kt:110~120

```
val dcRequest =
  Request(
    responseType = "vp_token",
    responseMode = "dc_api",
    nonce = nonce,
    dcqlQuery = dcqlQuery,
    transactionData = listOf(encodedTransactionData),
    clientMetadata = clientMetadata,
  )

val dpcRequest = DpcRequest(protocol = "openid4vp-v1-unsigned", request = dcRequest)
```

- EUDI Wallet 표준의 핵심: EUDI Wallet 은 OpenID4VP 를 주요 프로토콜로 사용합니다.
- Protocol = “openid4vp-v1-unsigned” - EUDI Wallet 도 동일한 프로토콜을 사용하여 Verifiable Presentation 을 요청합니다
- responseType = “vp_token”

3.9.2 ISO/IEC 18013-5 mode 형식 지원

예시 2: DpcHelper.kt:85~91

```
val credentialQuery =
    CredentialQuery(
        id = credId,
        format = mdocIdentifier,
        meta = Meta(doctypeValue = "com.emvco.payment_card"),
        claims = claims,
)
```

예시 3: DpcHelper.kt:96~100

```
val mdocFormatsSupported =
    MdocFormatsSupported(
        issuerauthAlgValues = listOf(-7), // ES256
        deviceauthAlgValues = listOf(-7),
    )
```

- EUID Wallet 의 주요 형식: EUID Wallet 은 ISO mdoc (mDL - Mobile Driver’ s License) 형식을 핵심 credential 형식으로 사용합니다
- Format = mdocIdentifier (mso_mdoc) - EUID Wallet 에서도 동일한 mdoc 형식을 사용합니다
- ES256 알고리즘 - EUID Wallet 에서 권장하는 암호화 알고리즘입니다

3.9.3 DCQL (Digital Credentials Query Language) 사용

예시 4: DpcHelper.kt:78~93

```
  // Build the DCQL query to request specific credential claims.
  val claims =
    listOf(
      Claim(path = listOf("com.emvco.payment_card.1", "card_number")),
      Claim(path = listOf("com.emvco.payment_card.1", "holder_name")),
    )

  val credentialQuery =
    CredentialQuery(
      id = credId,
      format = mdocIdentifier,
      meta = Meta(doctypeValue = "com.emvco.payment_card"),
      claims = claims,
    )
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
val dcqlQuery = DcqlQuery(credentials = listOf(credentialQuery))
```

- 선택적 공개 (Selective Disclosure) - EUDI Wallet 의 핵심 원칙입니다
- DCQL 을 사용하여 필요한 특정 claim 만 요청합니다 (카드 번호, 소유자 이름 만)
- EUDI Wallet 도 동일한 방식으로 사용자가 공개할 정보를 선택할 수 있습니다

3.9.4 Android Credential Manager API 통합

예시 5: DpcHelpper.kt:67~76

```
// Build transaction_data payload.
val transactionData =
  TransactionData(
    type = "payment_card",
    credentialIds = listOf(credId),
    transactionDataHashesAlg = listOf("sha-256"),
    merchantName = merchantName,
    amount = "US ${String.format("%.2f", totalValue)}",
    additionalInfo = json.encodeToString(additionalInfo), // Serialize the inner object
  )
```

- Android Credential Manager API 는 EUDI Wallet 의 구현 플랫폼 중 하나입니다
- Transaction Data 에 대한 서명 - EUDI Wallet 에서도 중요한 보안 메커니즘입니다
- transactionDataHashesAlg = “sha-256” - 거래 무결성을 보장합니다

예시 6: DpcHelper.kt:59~65

```
val additionalInfo =
  AdditionalInfo(
    title = "Please confirm your purchase details...",
    tableHeader = listOf("Name", "Qty", "Price", "Total"),
    tableRows = tableRows,
    footer = footerText,
  )
```

- EUID Wallet 의 핵심 원칙: 사용자가 공유하는 정보를 명확히 보고 동의해야 합니다
- 구매 세부사항을 표시하여 사용자가 서명하는 내용을 정확히 이해할 수 있게 합니다

3.9.5 DPC vs EUDI Wallet 차이점

- **용도**: DPC는 결제에 특화 (payment_card), EUDI Wallet은 신원 증명이 주요 목적
- **doctype**: com.emvco.payment_card vs EUDI Wallet의 eu.europa.ec.eudi.pid.1 (Personal ID)
- **서명**: 현재는 unsigned 버전을 사용하지만, EUDI Wallet은 강력한 암호화 서명 요구

3.9.6 DPC 한계점 및 고려사항

현재의 제약사항:

- **플랫폼 한정**: 현재 Android에만 구현, iOS 지원 제한적
- **생태계 미성숙**: CMWallet 등 credential provider가 아직 초기 단계
- **사용자 교육**: 새로운 개념으로 사용자 이해와 신뢰 구축 필요
- **표준화 진행 중**: OpenID4VP, mDOC 등이 아직 완전히 확립되지 않음

보안 및 개인정보:

- **검증 메커니즘**: 현재 샘플은 unsigned 버전, 실제 운영에서는 완전한 서명 검증 필수
- **키 관리**: 사용자 기기에서의 안전한 키 저장 및 관리 중요
- **재생 공격 방지**: Nonce 및 타임스탬프 검증 필요
- **프라이버시 역설**: 강력한 인증이 익명성과 상충될 수 있음

기술적 고려사항:

- **성능**: 암호학적 연산으로 인한 지연 시간 (일반적으로 1-2초)
- **저장 공간**: mDOC 형식의 credential 저장에 필요한 공간
- **네트워크 의존성**: 초기 credential 발급 시 온라인 연결 필요
- **호환성**: 다양한 Android 버전 및 기기 지원 필요

사업적 고려사항:

- **가맹점 수용**: 새로운 결제 방식에 대한 가맹점 시스템 업그레이드 필요
- **규제 준수**: 각국의 금융 규제 및 데이터 보호법 준수
- **비즈니스 모델**: 기존 카드 네트워크와의 수익 모델 차이
- **사용자 인센티브**: 복잡한 새 시스템 도입을 정당화할 혜택 제공 필요

3.10 DPC 개선 방향 (Next Steps)

3.10.1 서명된 OpenID4VP 흐름으로의 전환

현재 샘플은 `openid4vp-v1-unsigned` 프로파일을 사용하므로, Merchant Agent 는 토큰 진위 여부를 검증하지 못합니다. ARF(Security Level High)와 EUDI Wallet 파일럿에서는 서명/검증이 필수 요소로 간주되므로 다음 작업이 필요합니다.

- **서명 프로파일 채택**: EMVCo DC-API 서명 확장 또는 OpenID4VP `signed` 프로파일로 업그레이드하고, mdoc 기반 credential 서명을 해석할 수 있는 검증 로직을 Merchant Agent 에 추가합니다.
- **검증 서비스 분리**: 지갑에서 반환한 VP 토큰에 대해 issuer 서명, credential 유효기간, nonce 를 검증하는 Verifier/Wallet Back-End를 구성합니다.
- **Trust Framework 정합성**: ARF 가이드라인에서 요구하는 신뢰 anchor(국가/회원국 발급자 루트 인증서)와 Revocation 검사를 통합합니다.

3.10.2 재현 공격 및 키 관리 개선

- **Replay 방지**: nonce·session binding을 강화하고 Merchants/logger 쪽에서 사용된 토큰을 기록하여 재사용을 차단합니다.
- **지갑 키보호**: Android 하드웨어 백업 키 저장소(Keystore) 및 향후 iOS Secure Enclave를 이용한 키 저장/attestation을 도입합니다.

3.10.3 Merchant 측 보증 수준 상향

- **거래 영수증**: Merchant Agent 가 검증 결과와 고객 동의 evidence를 보관해 non-repudiation을 달성합니다.
- **Fallback 흐름**: 증명 실패 시 기존 카드 결제 혹은 스텝업 인증으로 전환하는 사용자 경험을 설계합니다.

3.10.4 iOS 및 기타 플랫폼 정합성

- **iOS 지원 현실화**: WWDC24에서 공개된 IdentityCredential / Digital Credentials API는 개발자 프리뷰 수준이므로, Apple의 Public Release 일정을 추적하고 베타 프레임워크 실험을 병행합니다.
- **프로토콜 정렬**: 현재 iOS 프레임워크는 ISO 18013-5/7 기반 mdoc 흐름만 공식 지원 예정으로 알려져 있으므로, OpenID4VP support roadmap 공개 시까지는 SD-JWT VC 또는 web-based bridge 방식을 병행합니다.
- **플랫폼 추상화**: Credential Manager (Android)·IdentityCredential(iOS)·WebAuthn 기반 브라우저 API를 아우르는 공통 추상화 레이어를 설계하여, 지갑/발급자/검증자 역할을 ARF와 동일한 구조로 매핑합니다.

3.10.5 생태계 및 규제 대응

- **규제 일치**: EMVCo DC-API, ETSI TS 119 495(수신자 서명) 등 결제용 표준을 모니터링하고, EUDI Wallet 시행규정(2024/1183)에서 요구하는 PID/EAA(전자 속성) 처리 절차를 문서화합니다.
- **인증 준비**: 지갑·검증자 모듈을 EUDI Wallet Conformity Assessment (BR-EL, DR-EL) 요구사항에 맞춰 테스트하고, 파일럿 참여국(예: EWC, NOBCCS)에서 공개한 상호운용성 체크리스트를 반영합니다.

CHAPTER 4

x402 시나리오

This illustrates the complete flow of the Human Present x402 Payment scenario using the AP2 framework with the A2A protocol and x402 payment standard.

4.1 Sequence Diagram

4.2 Key Components

4.2.1 1. IntentMandate Structure

The IntentMandate captures the user's shopping intent:

```
{
  "intent_mandate_id": "uuid",
  "natural_language_description": "I want to buy a coffee maker",
  "user_prompt_required": true,
  "merchants": ["merchant_agent"],
  "skus": ["coffee-maker-001"],
  "intent_expiry": "2025-11-18T10:00:00Z",
  "requires_refundability": false
}
```

4.2.2 2. CartMandate Structure with x402 Support

The CartMandate contains product and x402 payment request details:

```
{
  "cart_mandate_id": "uuid",
  "merchant_name": "Example Merchant",
  "cart_expiry": "2025-11-18T10:00:00Z",
  "refund_period": "P30D",
  "payment_request": {
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
    "method_data": [{
      "supported_methods": ["x402"],
      "data": {
        "supported_chains": ["ethereum", "polygon", "base"],
        "supported_currencies": ["USDC", "USDT", "DAI"],
        "merchant_wallet_address": "0x1234567890abcdef..."
      }
    }],
    "details": {
      "id": "order-123",
      "total": {
        "label": "Total",
        "amount": {"currency": "USDC", "value": "79.99"}
      },
      "displayItems": [
        {"label": "Coffee Maker", "amount": {"value": "69.99"}},
        {"label": "Shipping", "amount": {"value": "5.00"}},
        {"label": "Tax", "amount": {"value": "5.00"}},
        {"label": "Estimated Gas Fee", "amount": {"value": "0.50"}}
      ]
    }
  },
  "merchant_signature": "signature_by_merchant"
}
```

4.2.3 3. PaymentMandate Structure for x402

The PaymentMandate authorizes the x402 payment:

```
{
  "payment_mandate_contents": {
    "payment_mandate_id": "uuid",
    "timestamp": "2025-11-17T12:00:00Z",
    "payment_details_id": "order-123",
    "payment_details_total": {
      "label": "Total",
      "amount": {"currency": "USDC", "value": "79.99"}
    },
    "payment_response": {
      "request_id": "order-123",
      "method_name": "x402",
      "details": {
        "token": {
          "value": "encrypted_x402_token",
          "url": "http://localhost:8002/a2a/credentials_provider"
        },
        "chain_id": "1",
        "currency": "USDC",
        "contract_address": "0xA0b86991c6218b36c1d19D4a2e9Eb0cE3606eB48"
```

(다음 페이지에 계속)

(이전 페이지에서 계속)


```
      },
      "shipping_address": {
        "streetAddress": "123 Main St",
        "city": "San Francisco",
        "state": "CA",
        "zipCode": "94102"
      },
      "payer_email": "user@example.com"
    },
    "merchant_agent": "merchant_agent"
  },
  "user_authorization": "cart_hash_payment_hash_signature"
}
```


4.2.4 4. x402 Payment Credential Token

The token contains blockchain wallet credentials:


```
{
  "value": "encrypted_x402_payment_token",
  "url": "http://localhost:8002/a2a/credentials_provider"
}
```


The Credentials Provider decrypts this to reveal:


```
{
  "wallet_address": "0xabcdef1234567890...",
  "chain_id": "1",
  "currency": "USDC",
  "balance": "1000.00",
  "signing_authority": "delegated_signer_key",
  "is_smart_wallet": true
}
```


4.2.5 5. Blockchain Transaction Structure

The transaction submitted to the blockchain:


```
{
  "from": "0xabcdef1234567890...",
  "to": "0x1234567890abcdef...",
  "value": "0",
  "data": "0xa9059cbb...",
  "chainId": 1,
  "nonce": 42,
  "gasLimit": "100000",
  "maxFeePerGas": "50000000000",
  "maxPriorityFeePerGas": "2000000000"
}
```

4.2.6 6. Transaction Receipt

The blockchain receipt confirming the transaction:

```
{
  "transactionHash": "0x123abc...",
  "blockNumber": 12345678,
  "blockHash": "0x789def...",
  "from": "0xabcdef1234567890...",
  "to": "0xA0b86991c6218b36c1d19D4a2e9Eb0cE3606eB48",
  "gasUsed": "65000",
  "effectiveGasPrice": "50000000000",
  "status": "success"
}
```

4.3 기술 상세

4.3.1 HTTP 402 상태 코드

```
HTTP/1.1 402 Payment Required
Content-Type: application/json
WWW-Authenticate: Bearer realm="payment-api"

{
  "amount": "0.01",
  "currency": "USD",
  "payment_methods": ["crypto", "micropayment", "card"],
  "payment_endpoint": "https://payment.example.com/pay",
  "resource_id": "api-call-12345"
}
```

4.3.2 에이전트 자동 결제 로직

```
async def fetch_with_payment(url: str, agent_wallet):
    response = await http_client.get(url)

    if response.status_code == 402:
        payment_info = response.json()

        # 자동 결제 판단 로직
        if agent_wallet.can_afford(payment_info['amount']):
            # 결제 실행
            payment_token = await agent_wallet.pay(
                amount=payment_info['amount'],
                endpoint=payment_info['payment_endpoint']
            )

            # 결제 증명과 함께 재요청
            response = await http_client.get(
                url,
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
                headers={'Authorization': f'Bearer {payment_token}'}
            )

    return response
```

4.3.3 결제 토큰 검증

```
def verify_payment_token(token: str, resource_id: str):
    try:
        decoded = jwt.decode(token, public_key, algorithms=['RS256'])

        # 토큰 유효성 확인
        assert decoded['resource_id'] == resource_id
        assert decoded['exp'] > time.time()
        assert decoded['amount_paid'] >= required_amount

        return True
    except:
        return False
```

CHAPTER 5

BOSH Agent Development Setup

```
PATH=.../s3cli/out:$PATH bin/run -P ubuntu -C path_to_config.json
```

5.1 Running locally

To start server locally:

```
gem install nats
nats-server
```

To subscribe:

```
nats-sub '>' -s nats://localhost:4222
```

To publish:

```
nats-pub agent.123-456-789 '{"method":"apply","arguments":[{"packages":[{"name":"package-
↪name", "version":"package-version"}]}]}' -s nats://localhost:4222
```

5.2 Blobstores

The Go Agent ships with 4 default blobstores:

- Local filesystem
- S3
- DAV
- Dummy (for testing)

You can, however, use custom blobstores by implementing a simple interface. For example, if you want to use a blobstore named "custom" you need to create an executable named

`bosh-blobstore-custom` somewhere in `PATH`. This executable must conform to the following command line interface:

- `-c` flag that specifies a config file path (this will be passed to every call to the executable)
- must parse the config file in JSON format
- must respond to `get <blobID> <filename>` by placing the file identified by the blobID into the filename specified
- must respond to `put <filename> <blobID>` by storing the file at filename into the blobstore at the specified blobID

A full call might look like:

```
bosh-blobstore-custom -c /var/vcap/bosh/etc/blobstore-custom.json get 2340958ddfg /tmp/my-
↪cool-file
```

5.3 Set up a workstation for development

Note: This guide assumes a few things:

- You have gcc (or an equivalent)
- You can install packages (brew, apt-get, or equivalent)

Get Golang and its dependencies (Mac example, replace with your package manager of choice):

- `brew update`
- `brew install go`
- `brew install git` (Go needs git for the `go get` command)
- `brew install hg` (Go needs mercurial for the `go get` command)

Clone and set up the BOSH Agent repository:

- `go get -d github.com/cloudfoundry/bosh-agent`
    - Note that this will print an error message because it expects a single package; our repository consists of several packages. The error message is harmless—the repository will still be checked out.
- `cd $GOPATH/src/github.com/cloudfoundry/bosh-agent`

From here on out we assume you're working in `$GOPATH/src/github.com/cloudfoundry/bosh-agent`

Install tools used by the BOSH Agent test suite:

- `bin/go get code.google.com/p/go.tools/cmd/vet`
- `bin/go get github.com/golang/lint/golint`

5.3.1 Updating dependencies

All dependencies:

```
./update-dep github.com/cloudfoundry/bosh-utils/...
```

An example for a single dependency:

```
./update-dep github.com/pivotal-golang/clock/fakeclock
```

The script will pull latest package version and copy its files to vendor directory. Note, that it does not pull dependecies recursively from package unless they are vendored.

For development purposes, delete package from vendor use it from your $GOPATH. Once everything works in other package push changes in that package to github and run ./update-dep to add latest version.

Every vendored packaged is recorded in deps.txt with git/hg sha of the package for references. The known issue if subpackages are being vendored and later full package is being vendored subpackage references are not deleted in deps.txt.

To delete package `rm -rf vendor/path_to_package` and delete reference in deps.txt.

5.3.2 Running tests

Each package in the agent has its own unit tests. You can run all unit tests with `bin/test-unit`.

Additionally, BOSH[5] includes integration tests that use this agent. Run `bin/test-bosh-integration` to run those with your local agent changes. However, in order to run the BOSH integrations tests, you will need a copy of the BOSH repo, which this script will do in `./tmp`. BOSH uses Ruby for its tests, so you will also need to have that available.

You can run all the tests by running `bin/test`.

There is a stand-alone BOSH Agent integration test to test config-drive using bosh-lite. This can be run locally via `bin/test-integration --provider=virtualbox`. The vagrant provider passed in can be changed to a provider of your choosing if you so desire.

5.3.3 Using IntelliJ with Go and the BOSH Agent

- Install IntelliJ 15[6]
- Install the latest Google Go plugin for IntelliJ[7]. You may want to grab the latest early access (EAP) build, rather than the last release.
- (Optional) Download, Install & Select improved keybindings[8] for IntelliJ:
  - `git clone git@github.com:Pivotal-Boulder/IDE-Preferences.git`
  - `cd ~/Library/Preferences/IntelliJIdea13/keymaps`
  - `ln -sf ~/workspace/IDE-Preferences/IntelliJKeymap.xml`
  - In IntelliJ: Preferences -> Keymap -> Pick 'Mac OS X 10.5+ Improved'

[5] https://github.com/cloudfoundry/bosh
[6] http://www.jetbrains.com/idea/download/index.html
[7] https://github.com/go-lang-plugin-org/go-lang-idea-plugin
[8] https://github.com/Pivotal-Boulder/IDE-Preferences

- Clone bosh-agent into a clean go workspace (or use a bosh[9] clone with bosh/go as the workspace root):
  - `mkdir -p ~/workspace/bosh-agent-workspace/src/github.com/cloudfoundry`
  - `cd ~/workspace/bosh-agent-workspace/src/github.com/cloudfoundry`
  - `git clone https://github.com/cloudfoundry/bosh-agent`
- Open ~/workspace/bosh-agent-workspace as a new project in IntelliJ.
- Set the Go SDK as the Project SDK:
  - Open the Project Structure window: `File -> Project Structure`
  - Select the `Project` tab in left sidebar
  - (Optional) Add a `New` Go SDK by selecting your go root.
  - Select `Go SDK go1.3` under Project SDK
- Setup module sources
  - Open the Project Structure window: `File -> Project Structure`
  - Select the `Modules` tab in left sidebar
  - Select your module in the middle sidebar
  - Select the `Sources` tab in the Module pane
  - Select ~/workspace/bosh-agent-workspace/src and add is as a source dir
- Install & configure the Grep Console[10] plugin
  - Install via `Preferences -> Plugins`
  - Select `Preferences -> Grep COnsole -> Enable ANSI coloring` to colorize Ginkgo test output
- Re-index your project: `File -> Invalidate Cache / Restart`

You should now be able to 'go to declaration', auto-complete, and run tests from within IntelliJ.

[9] https://github.com/cloudfoundry/bosh
[10] https://github.com/krasa/GrepConsole

CHAPTER 6

BOSH linux stemcell builder

Repository: https://github.com/cloudfoundry/bosh-linux-stemcell-builder

기본적으로 프로젝트에 Vagrantfile 이 포함되어 있기 때문에 build 에 필요한 환경을 만들기 위해서는 VirtualBox 와 Vagrant 가 깔려 있으면 된다. 하지만 이를 자동화된 방식으로 반복하기 위해서는 CI 를 활용하는게 나은 방법일 것이고 다음은 Cuncourse CI 를 통한 Build 절차이다.

6.1 Concourse workflow

To create a stemcell on concourse instead of locally on virtualbox, you can execute the build-stemcell task.

```
mkdir /tmp/version
cat <<EOF >/tmp/version/number
0.0
EOF
cd /tmp/version
git init

pushd ~/workspace/bosh-linux-stemcell-builder
fly -t production login
IAAS=vsphere HYPERVISOR=esxi OS_NAME=ubuntu OS_VERSION=trusty time fly -t production
↪execute -p -x -i version=/tmp/version -i bosh-linux-stemcell-builder=. -c ./ci/tasks/
↪build.yml -o stemcell=/tmp/vsphere/dev/
popd
```

Q: 내 local /tmp/version 에 version 정보가 관리되어야 하는가? 지워지면 어떻하지? 버전번호 업데이트는 알아서 해주나?

6.2 Periodic bump

OS images are stored in S3 bucket bosh-os-images[11].

stemcell 에 사용되는 OS Image 는 항상 중요 보안 문제와 심각한 버그들로 부터 안전하게 시스템이 지켜지도록 자동화된 방식으로 지속 업데이트 된다. 특별한 수정사항이 발생한 업데이트의 경우 패치 내용이 기술되지만 정기적인 자동 업데이트의 경우는 Periodic bump 로 기술된다.

6.3 Rakefile

참조: https://github.com/cloudfoundry/bosh-linux-stemcell-builder

프로젝트 최상단에 위치한 Rakefile 은 Stemcell Build Process 의 전체 구조를 살펴보는데 도움이 될 것이다. Rakefile 에 의하면 총 4 단계를 거쳐 stemcell 이 제작되는 것을 알 수 있다. 만약 S3 와 같은 원격 저장소를 활용하지 않는다면 이를 2 단계로 간소화 할 수 있다.

Rakefile 에 기술된 stemcell build process

1. build_os_image
2. upload_os_image (optional)
3. download_os_image (optional)
4. build(_with_local_os_image)

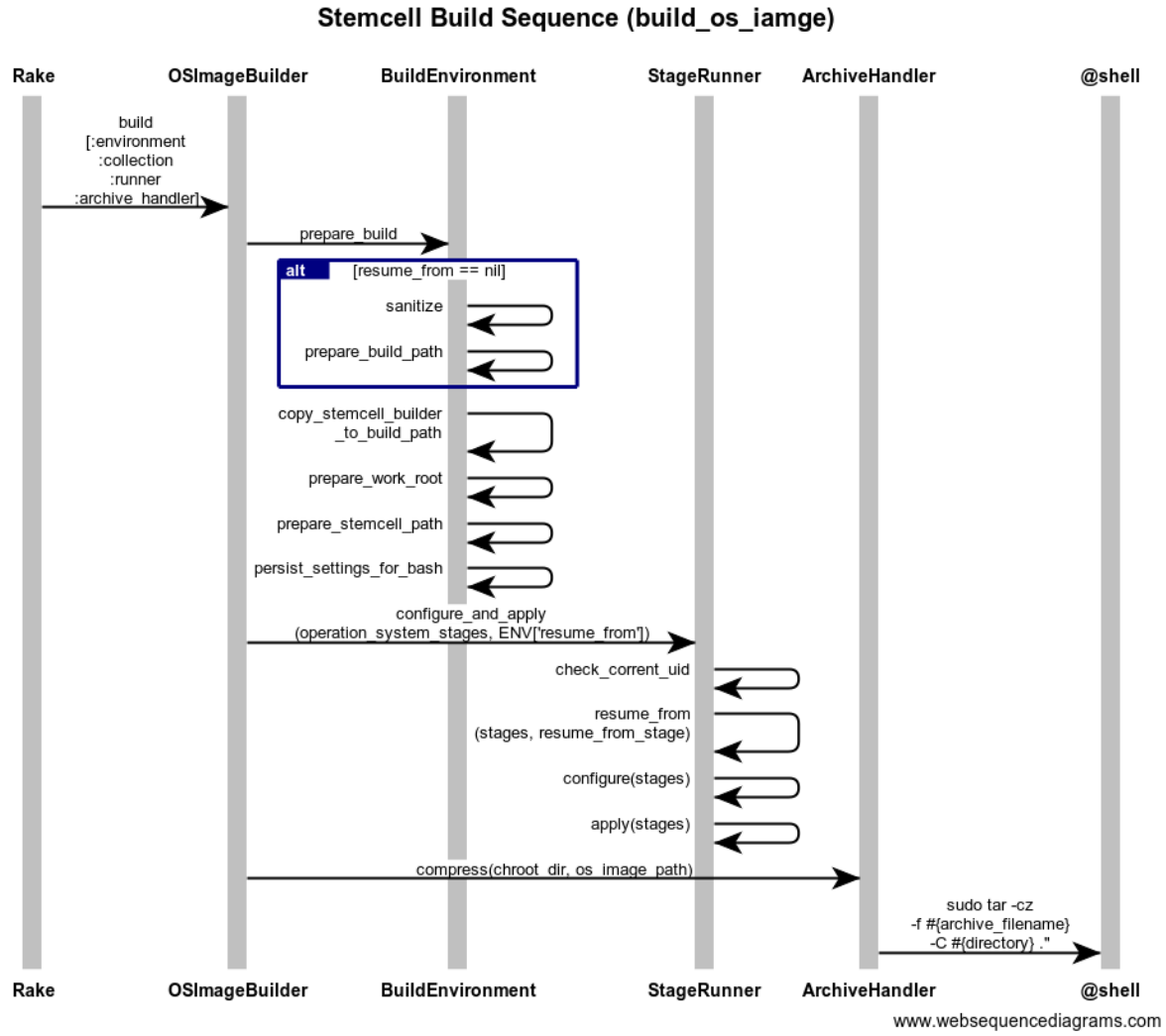
S3 를 사용하지 않을때의 stemcell build process

1. build_os_image
2. build_woth_local_os_image

6.4 build_os_image

stemcell 의 모체가 되는 OS Image 를 생성한다. 옵션에 따라 Ubuntu / CentOS / ReadHat 등을 선택할 수 있으며 Public 으로 배포되는 ISO Image 에 약간의 hardning 작업을 가해 보안성을 강화하고 bosh 와 관련된 모듈들을 추가 설치한 Image File 이 생성된다

[11] http://s3.amazonaws.com/bosh-os-images/




```
title Stemcell Build Sequence (build_os_iamge)
Rake->OSImageBuilder:build\n[:environment\n:collection\n:runner\n:archive_handler]
OSImageBuilder->BuildEnvironment:prepare_build
    alt resume_from == nil
        BuildEnvironment->BuildEnvironment:sanitize
        BuildEnvironment->BuildEnvironment:prepare_build_path
    end
    BuildEnvironment->BuildEnvironment:copy_stemcell_builder\n_to_build_path
    BuildEnvironment->BuildEnvironment:prepare_work_root
    BuildEnvironment->BuildEnvironment:prepare_stemcell_path
    BuildEnvironment->BuildEnvironment:persist_settings_for_bash
OSImageBuilder->StageRunner:configure_and_apply\n(operation_system_stages, ENV['resume_from
↪'])
    StageRunner->StageRunner:check_corrent_uid
    StageRunner->StageRunner:resume_from\n(stages, resume_from_stage)
    StageRunner->StageRunner:configure(stages)
    StageRunner->StageRunner:apply(stages)
OSImageBuilder->ArchiveHandler:compress(chroot_dir, os_image_path)
    ArchiveHandler->@shell:sudo tar -cz \n-f #{archive_filename} \n-C #{directory} ."
```

stemcell 은 StageCollection class 에 정의된 stages 라고 불리우는 묶음을 StateRunner class

를 통해 순차적으로 수행함으로써 OS 이미지를 생성한다. OSImageBuilder 가 StageRunner 를 초기화 할때 StageCollection.operation_system_stages 를 전달하므로 이 states 를 순차적으로 수행할 것이다. IaaS/OS 별로 서로 다른 stages 가 정의되어 있으며 상황에 맞는 state 묶음인 stages 를 정의해 둔 소스가 다음 위치에 있다.

```
bosh_stemcell/lib/bosh/stemcell/state_collection.rb
```

Stage 는 'apply.sh' 에 의해 실행되며 100% 독립적으로 실행될 수 있도록 Stage 간에 의존성이 있으면 안된다. states/ 디렉터리의 하부는 다음과 같이 이루어져 있다.

- apply.sh state 실행 스크립트
- assets/* 각 asset 은 stage 를 수행하기 위해 필요한 모든 resource 들이 보관되는 장소이다.
- config.sh stage 내에서 필요한 local configuration 저장소.

StageCollection.operation_system_stages 를 분석하여 Ubuntu OS Image 를 위한 stages 는 다음과 같이 구성되어 있음을 알게 되었다.

1. base_debootstrap,
2. base_ubuntu_firstboot,
3. base_apt,
4. base_ubuntu_build_essential,
5. base_ubuntu_packages,
6. base_file_permission,
7. base_ssh,
8. bosh_sysstat,
9. system_kernel,
10. system_kernel_modules,
11. system_ixgbevf (Arch.ppc64le 인 경우 생략)
12. **bosh_steps**
    1. bosh_sysctl,
    2. bosh_limits,
    3. bosh_users,
    4. bosh_monit,
    5. bosh_ntpdate,
    6. bosh_sudoers,
13. password_policies,
14. restrict_su_command,
15. tty_config,
16. rsyslog_config,
17. delay_monit_start,

18. system_grub,
19. vim_tiny,
20. cron_config,
21. escape_ctrl_alt_del,
22. system_users,
23. bosh_audit_ubuntu,
24. bosh_log_audit_start,

6.5 upload_os_image / download_os_image

build_os_image 에 의해 생성된 image 를 upload/download 한다. 한번 만들어 둔 것을 upload 해 두면 stemcell 제작시 재활용 할 수 있게 된다. local_os_image 를 사용한다면 필요하지 않다.

6.6 build(_with_local_os_image)

upload_os_image 에 의해 저장된 원격 저장소를 이용하지 않는다면 build_with_local_os_image 를 사용하면 된다. 만들어진 OS Image 를 Bosh 가 인식할 수 있는 stemcell 이라는 형태로 한번 더 감싸는 작업을 하게 된다.

우선, stages 는 몇개의 큰 카테고리로 분류된다.

- initialize
- operating_system_stages
- extract_operating_system_stages
- agent_stages
- build_stemcell_image_stages
- package_stemcell_stages

또한 StageCollection class 는 Forwardable 이어서 다음 class 들로 적절한 위임이 일어난다.

- definition
- infrastructure
- operating_system
- agent

이제 이렇게 분류해 놓은 stages 는 StemcellBuilder 와 StemcellPackager 에 의해 build 되는데 build 순서는 다음과 같다.

1. StemcellBuilder
    1. extract_operating_system_stages
        1. untar_base_os_image
    2. agent_stages
        1. bosh_libyaml
        2. bosh_go_agent

        3. aws_cli
        4. logrotate_config
        5. dev_tools_config
        6. static_libraries_config
    3. build_stemcell_image_stages
        1. system_network
        2. system_openstack_clock
        3. system_openstack_modules
        4. system_parameters
        5. bosh_clean
        6. bosh_harden
        7. bosh_openstack_agent_settings
        8. bosh_clean_ssh
        9. image_create
        10. image_install_grub
        11. finish_stemcell_stages
            1. bosh_package_list
2. StemcellPackager
    1. package_stemcell_stages
        1. raw_package_stages
            1. prepare_raw_image_stemcell
    2. write_manifest
    3. create_tarball

CHAPTER 7

BOSH lite resizing

7.1 resizing memory

최초 시작시 (아직 VM Provisioning 이 수행되기 전) Vagrantfile 을 열어 config.vm.provider 내에 v.memory 를 MB 단위로 지정하고 vagrant up 을 하면 메모리가 조정된다.

```
Vagrant.configure('2') do |config|
  config.vm.box = 'cloudfoundry/bosh-lite'

  config.vm.provider :virtualbox do |v, override|
    override.vm.box_version = '9000.137.0' # ci:replace
    # To use a different IP address for the bosh-lite director, uncomment this line:
    # override.vm.network :private_network, ip: '192.168.59.4', id: :local
    v.memory = 40960
  end

  config.vm.provider :aws do |v, override|
    override.vm.box_version = '9000.137.0' # ci:replace
    # To turn off public IP echoing, uncomment this line:
    # override.vm.provision :shell, id: "public_ip", run: "always", inline: "/bin/true"

    # To turn off CF port forwarding, uncomment this line:
    # override.vm.provision :shell, id: "port_forwarding", run: "always", inline: "/bin/
↪true"

    # Following minimal config is for Vagrant 1.7 since it loads this file before
↪downloading the box.
    # (Must not fail due to missing ENV variables because this file is loaded for all
↪providers)
    v.access_key_id = ENV['BOSH_AWS_ACCESS_KEY_ID'] || ''
    v.secret_access_key = ENV['BOSH_AWS_SECRET_ACCESS_KEY'] || ''
    v.ami = ''
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
  end

  config.vm.provider :vmware_fusion do |v, override|
    override.vm.box = 'cloudfoundry/no-support-for-bosh-lite-on-fusion'
    #we no longer build current boxes for vmware_fusion
    #ensure that this fails. otherwise the user gets an old box
  end

  config.vm.provider :vmware_workstation do |v, override|
    override.vm.box = 'cloudfoundry/no-support-for-bosh-lite-on-workstation'
    #we no longer build current boxes for vmware_workstation
    #ensure that this fails. otherwise the user gets an old box
  end
end
```

7.2 resizing disk

디스크의 조정은 다음 문서를 참조했다. https://medium.com/@phirschybar/resize-your-vagrant-virtualbox-disk-3c0fbc607817

Vagrant 가 Provisioning 한 디스크는 vmdk 포맷인데 이 포멧은 VBoxManager 를 통해 동적으로 resizing 을 할 수 없다. 따라서 다음 과정을 통해 용량 증설이 가능하다.

- vmdk -> vdi 로 포맷 변경
- vdi resizing
- guest 에서 파티션 생성
- LVM 으로 새로 생성된 파티션을 기존 파티션에 붙이기

```
~/host> vagrant halt
~/host> cd /path/where/your/vbox/.vmdk/files/are
~/host> VBoxManage clonehd my.vmdk out.vdi --format VDI
~/host> VBoxManage modifyhd out.vdi --resize 51200
```

이렇게 하면 이미 떠있던 VM 이 halt 된다. 나중에 다시 VM 을 기동하여도 warden 컨테이너는 제대로 기동이 안되므로 deploy 를 수행하기 전에 용량 증설을 먼저 해두어야 한다.

그런 다음 VirtualBox GUI 를 통해 하드 드라이브를 새로 만든 .vdi 이미지로 바꿔치기 한다. (select VM > settings > storage > remove existing hard drive and add bew hard drive)

```
~/host> vagrant up
~/host> vagrant ssh
~/guest> sudo cfdisk /dev/sda
```

이렇게 하면 VM 이 기동되고 파티션 정보를 보여주는 대화형 창이 뜬다. 새로운 Space 를 선택후 다음과 같은 명령을 수행한다. (Select new space available > select “NEW” > select “Primary” > select “Write”) /sda3 가 새로운 파티션으로 나타날 것이다.

```
~/guest> exit
~/host> vagrant reload
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
~/host〉 vagrant ssh
~/guest〉 sudo pvcreate /dev/sda3
```

만약 pvcreate 명령이 설치되어 있지 않다면 **apt install lvm2** 로 설치힌다.

```
~/guest〉 sudo pvdisplay ¦ grep "VG Name"
```

이것은 볼륨 그룹 이름을 출력해 줄 것이다. 내 경우는 ubuntu-vg 였다. 다음과 같이 하여 볼륨 그룹에 새로 생성된 파티션을 붙여 확장한다. VG-NAME 을 자신의 것으로 바꿀것.

```
~/guest〉 sudo vgextend VG-NAME /dev/sda3
~/guest〉 sudo lvextend /dev/VG-NAME/root /dev/sda3
~/guest〉 sudo resize2fs /dev/VG-NAME/root
~/guest〉 df -h
```

마지막 명령은 새롭게 조정된 디스크 볼륨을 보여줄 것이다.

CHAPTER 8

Using upstart on ubuntu

8.1 오래 돌아야 하는 서버 또는 워커를 관리하는 스마트하지 않은 방법

- screen 이나 tmux 안에 띄워놓고 잊어버리기
- nohup 으로 실행해두고 잊어버리기

8.2 스마트하지 않은 방법의 문제점

- 프로세스가 꺼졌는지 한참동안 모르고 있다가 당황하기
- 시스템 재부팅 될때 두려움에 떨기

8.3 우분투에서 기본으로 제공되는 upstart 를 사용하는 스마트한 방법

- 시스템 부팅 시에 서비스 띄우기
- 다른 서비스가 시작된 후에 서비스 띄우기
- 프로세스가 오류로 꺼지면 자동으로 다시 띄우기
- stdout/stderr 를 로그 파일에 기록하기
- 로그 파일이 커지면 쪼개기

8.4 설정 파일 설치하기

upstart 서비스 설정 파일은 /etc/init 에 모여 있다. 따라서 /etc/init 디렉토리에 <서비스명>.conf 파일을 만들어 넣으면 된다.

8.5 심볼릭 링크로 설치하기

/etc/init 에는 시스템 서비스의 설정 파일도 모두 들어있으므로 관리의 용이성을 위해 별도의 디렉터리에 서비스 설정 파일을 분리해 보관하는 것이 권장됨. 분리된 설정은 심볼록 링크를 걸어서 같은 효과를 누릴 수 있다.

```
sudo ln -s /home/my/service/some.conf /etc/init/
```

Caution!! /etc/init/ 에 직접 파일을 만들었을 때와 달리 변경사항을 upstart 가 감지하지 못하므로 다음 명령을 통해 명시적인 재로딩을 수행해야 함.

```
sudo initctl reload-configuration
```

8.6 서비스 관리

- sudo start <service_name>
- sudo stop <service_name>
- sudo restart <service_name> (주의: 설정 파일을 다시 읽어오지 않음. 설정 변경시 stop/start 혹은 reload 할 것)
- sudo reload <service_name> (프로세스에 HUP 시그널을 보내는 것임

8.7 설정 파일 작성

8.7.1 명령어 지정

가장 간단한 방법은 exec 뒤에 명령얼르 쓰면 됨.

```
exec uname -a
```

긴 쉘 스크립트가 필요한 경우 script 구문을 쓰면 됨.

```
script
    sleep 5
        uname -a
end script
```

어떤 명령어들은 실행하면 자동으로 백그라운드로 돌아가는 (detach/daemonize) 경우가 있는데 이를 방지하는 옵션이 있다면 켜두는 것이 좋다 (보통 -foreground 혹은 -nodetach)

Foregroup mode 실행 옵션이 없는 경우 여기[12] 를 참고 해서 설정하시라.

8.7.2 자동 시작 / 중단 조건

서비스를 조건에 따라 시작되거나 중단되게 할 수 있다. 가장 흔히 사용하는 조건은 부팅시 시작, 종료시 중단일텐데 다음과 같이 적으면 됨.

```
start on runlevel [2345]
stop on runlevel [016]
```

[12] http://upstart.ubuntu.com/cookbook/#expect

만약, 특정 서비스가 시작된 이후에 띄우고 싶다면 다음과 같이 아라

```
start on started <other_service>
```

and 연산자로 여러 조건을 조합할 수도 있다.

```
start on started mysql and started nginx
```

8.7.3 실행 권한 설정

프로세스는 기본적으로 root 권한을 가지고 실행된다. 보안을 강화하기 위해 별도의 사용자/그룹 권한으로 실행하고자 할 경우 다음과 같이 한다.

```
setuid <username>
setgid <groupname>
```

8.7.4 실행 환경 설정

환경변수를 설정 (env KEY=vlaue) 한 경우에는 exec/script 구문 내에서 $KEY 로 참조할 수 있다.

디렉터리 변경은 다음과 같이 할 수 있다.

```
chdir /path/to/current/dir
```

8.7.5 자동 재시작 (respawn)

프로세스가 예기치 않게 종료 되었을 때 자동으로 재실행 하는 방법

```
respawn
```

프로세스가 너무 빨리 되살아나는 것을 방지하기 위해 5초 동안 10번 재시작이 되면 더이상 재시작 하지 않는다. 이 제한은 respawn limit 으로 바꿀수 있음

Caution!! respawn limit 이 설정되었더라도 respawn 구문이 없으면 자동 재시작이 되지 않는다.

```
respawn
respawn limit 5 10 # 10 초동안 5 번 재시작이 되면 포기한다.
respawn limit unlimited
```

8.7.6 로그 파일

stdout/stderr 로 출력된 내용은 /var/log/upstart/<service_name>.log 에 저장된다. 우분투 (14.4) 기준으로는 이 로그 파일이 하루에 한번 분할 되고 최대 7개의 파일이 유지되는 것이 기본 설정이다. /etc/logrotate.d/upstart 에서 설정을 바꿀 수 있다.

8.7.7 설정 파일 예제

gunicorn 으로 파이선 웹 어플리케이션을 실행하는 예제

```
start on runlevel [2345]
stop on runlevel [016]
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
respawn
setuid www-data
setgid www-data
env PORT=11000
chdir /home/ditto/animeta

exec env/bin/gunicorn animeta.wsgi --bind 127.0.0.1:$PORT --error-logfile -
```

8.8 Alternatives

1. 우분투 차기 버전에서 upstart 가 systemd[13] 로 대체될 예정이라고 함. 이미 Debian linux 에서는 systemd 가 기본으로 탑재되어 있다.
2. 시스템과 독립적으로 작동하면서 upstart 같은 기능을 제공하는 supervisor[14] 라는 것도 많이 쓰이고 있음

8.9 참조 자료

- http://blog.sapzil.org/2014/08/12/upstart/

[13] http://www.freedesktop.org/wiki/Software/systemd/
[14] http://supervisord.org/

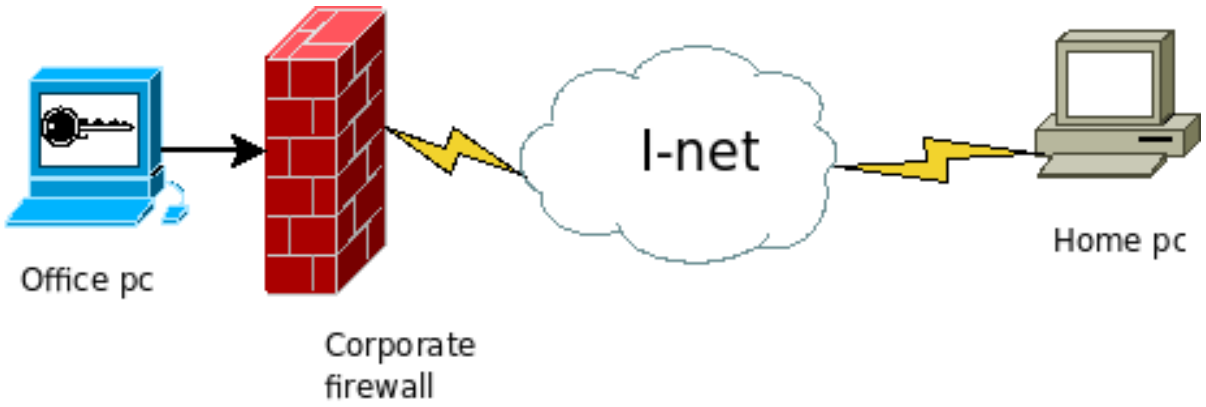
CHAPTER 9

Bypassing corporate firewall with reverse ssh port forwarding

From toic.org[15]

Probably lots of you are behind some sort of very restrictive corporate firewall. Unable to access your office pc from home because of firewall policies. In normal cases this scenario is more than welcomed. No outsiders should be allowed to access internal parts of secure network! Ideally companies will setup secure VPN access allowing its employees to access their work computers and do some work remotely. What if you aren' t one of the lucky ones having such option? You desperately need to access your office pc?

9.1 The problem



As shown on the picture above, we have our office PC behind very restrictive corporate firewall connected to Internet. Firewall will not allow any traffic originating from Internet to internal networ k except previously initiated traffic. Meaning you can contact remote hosts on Internet from your office PC and they can respond, but remote computers can' t initiate connection to your office PC. Thi s is of course huge problem if you have to access your work materials on office PC from your home. Additionally corporate firewall will only

[15] https://toic.org/blog/2009/reverse-ssh-port-forwarding

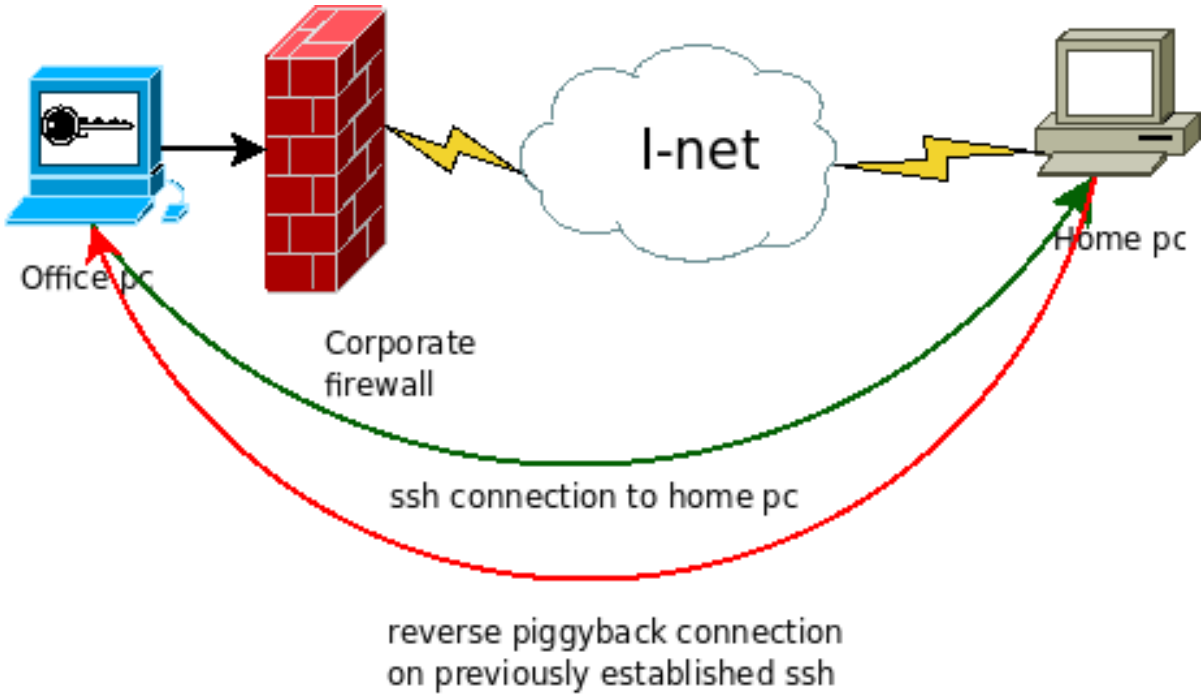
allow certain traffic from your office PC to remote hosts. Meaning you can only establish FTP, SSH, HTTP, POP3··· communications, all other ports are blocked.

9.1.1 How can you access your office PC then?

One way is to setup corporate VPN access allowing secure connections to internal network. Another method is to setup a port forwarding on corporate firewall so it redirects certain ports to your offi ce PC. But if you don' t have the means to accomplish any of this then the only way to do it is to use ssh tunnels and reverse port forwarding.

9.2 The solution

If we can only contact remote hosts on certain ports, the solution would be to contact remote hosts via allowed port and piggyback the connection on already established link.



Something like shown on the picture above. Fortunately we can do this with ssh.

9.3 Real life example

I will assume that home PC is connected via dynamically assigned IP address. First thing you will need to make sure you have ssh server installed on your home PC and it should be accessible from Inte rnet. If you have some NAT routers, be sure to forward port 22 to your home PC. Secondly you will need to setup a dyndns account so you can connect to your home PC regardless of IP address changes. N ow the goal will be to connect to that home ssh server from our office PC.

> For the purpose of this example i will name my home PC: bhome.dyndns.com Office computer name will be bwork.office.com bwork computer uses private IP range of 192.168.0.0/24 with address 192.168.0.100

So if the firewall is preventing outside connections to our bwork computer we must initiate connection from it.

We can do this with simple ssh command:

```
ssh -R 2210:localhost:22 bhome.dyndns.com
```

So what just happened here? We are initiating ssh connection **ssh** with reverse port forwarding option **-R** which will then open listening port **2210:** who is going to be forwarded back to **localhost**' s port **:22** an d all this will happen on remote computer **bhome.dyndns.com**.

This connection represents the green line in the diagram above, and it' s a legit connection as far as corporate firewall is concerned.

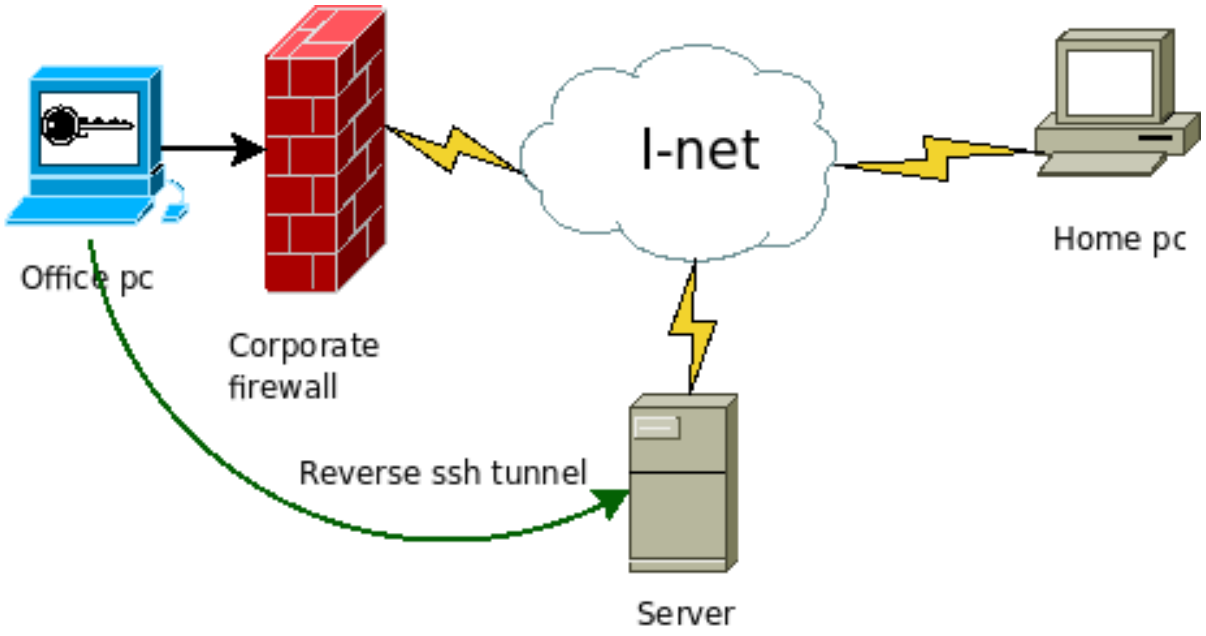
Now if weopen up a terminal on bhome computer, and type in:

```
ssh -p 2210 localhost
```

we will try to connect to **localhost** (bhome.dyndns.com) on port **2210**. Since that port is setuped by remote ssh connection it will tunnel the request back via that link to the **bwork.office.com** computer. This is the red line on the diagram above. Looking from firewall' s perspective it' s a legit traffic, since it is responding traffic on already initiated link from **bwork** computer.

9.4 Real life example 2

What if your home computer is not always on-line? Or perhaps you wish to access your office computer from multiple locations? For this you will have to have some dedicated server or VPS outside the c orporate firewall.



To accomplish this we will use the same command as previously, only this time we will open up a reverse ssh tunnel to remote server or VPS.

> For the purpose of this example we will name the server bserver.outside.com with IP 89.xxx.xx.4

```
ssh -R 2210:localhost:22 bserver.outside.com
```

again this will open up reverse ssh tunnel to the machine 89.xxx.xx.4 (bserver.outside.com). So when we login to the server and issue the command:

```
ssh -p 2210 localhost
```

we will end up with bwork computer' s ssh login prompt.

Can I use this previously established reverse ssh tunnel to the server to directly connect to my office computer?

Of course, but some slight modifications are required. By default ssh tunnels only bind to local address, and can be accessible only locally. Meaning, in the example above, you can' t just type:

```
ssh -p 2210 bserver.outside.com
```

on your home PC and be connected to your office PC!

If you run:

```
netstat -ntl
```

on bserver you will see that the port 2210 is only listening on 127.0.0.1 IP address. To get it listen on interface connected to Internet we must enable **GatewayPorts** option in ssh server' s config uration. By default GatewayPorts are disabled in sshd, but we can simply enable them:

```
vim /etc/ssh/sshd_config
```

then add:

```
GatewayPorts clientspecified
```

save the file and restart sshd:

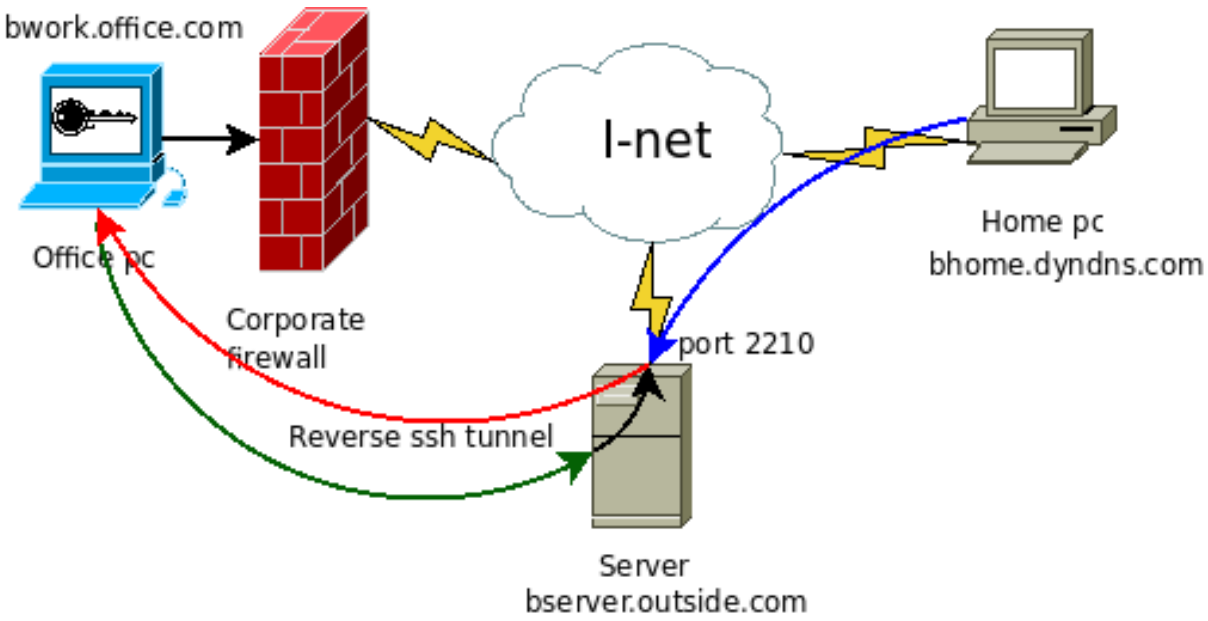
```
/etc/init.d/ssh restart
```

We could have just enable GatewayPorts by typing **On** instead of **clientspecified**, that would route any ssh tunnel to network interface. This way we can control which tunnel will be accessible f rom outside, and on which interface.

So if we initiate reverse ssh tunnel like this:

```
ssh -R 89.xxx.xx.4:2210:localhost:22 bserver.outside.com
```

we will have bserver listening on port 2210 on network interface bound to ip 89.xxx.xx.4 and forwarding all traffic via established tunnel to bwork computer. If you omit the 89.xxx.xx.4 address from the command above server will again listen on port 2210 only on local loopback interface. If you have multiple network interfaces on server be sure to select the one you can connect to. To enable listening port on all interfaces, just use IP 0.0.0.0 in the command above



Now when we run:

```
ssh -p 2210 bserver.outside.com
```

from our home PC we will initiate ssh connection on port **2210** towards server **bserver.outside.com** (blue line). Server will then forward that traffic to office PC (red line) via the previously established reverse ssh tunnel (green line).

> Of course you will have to open up port 2210 on server' s firewall to be able to connect.

9.5 Some more fun with reverse tunnels

Fun doesn' t stops there. Say I have a printer behind that corporate firewall. How can i connect to it? Easy··· remember the first example? the command ssh -R is taking 5 arguments of which 4 are mandatory:

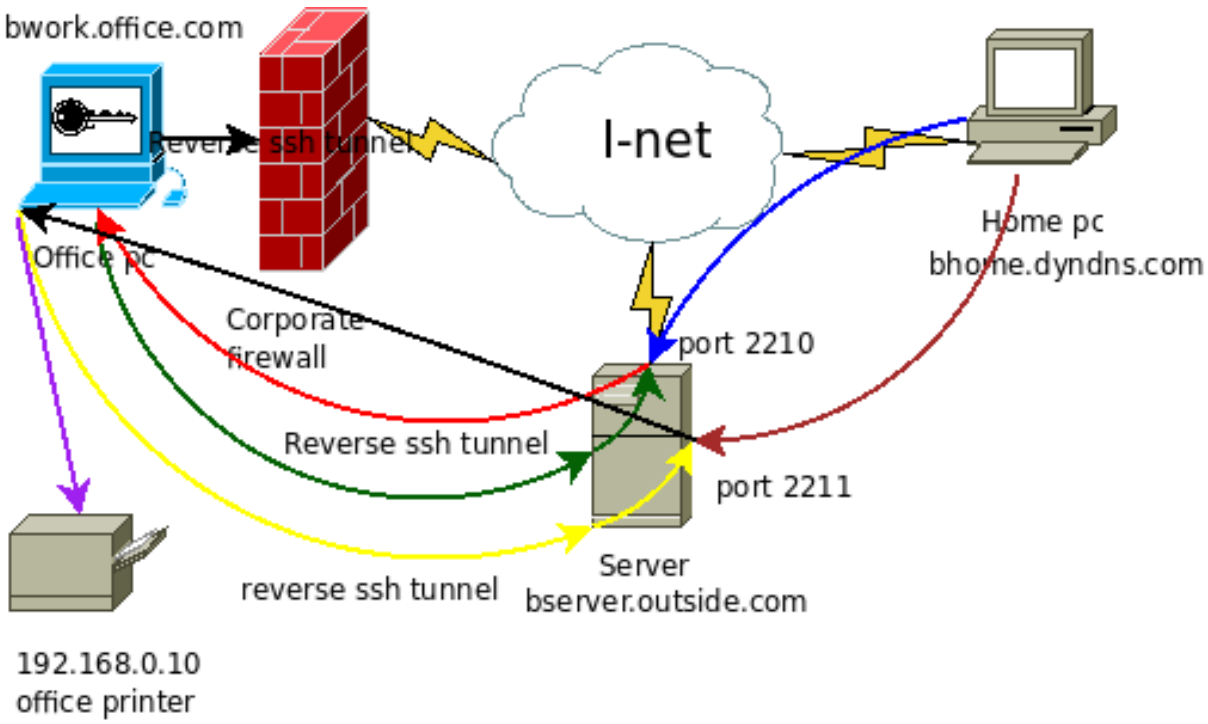
```
ssh -R [bind_address:]port:host:hostport
```

bind_address is the network address on which **port** will be listening, and forwarded to **host** (connected to network from which reverse tunnel originated) on **hostport**.

so if we issue the command like this on our bwork pc:

```
ssh -R 89.xxx.xx.4:2211:192.168.0.10:631 bserver.outside.com
```

we will get something like this:



so again we have previously established reverse ssh tunnel listening on port **2210** to channel the ssh connection towards office PC. Now with this new command we established the reverse ssh tunnel (yellow line) towards bserver which will listen for incoming connections on port **2211**. When the home pc makes a data connection to port **2211** on bserver (brown line) it is t hen forwarded to office PC (**black line**) which is then redirected towards office printer at address **192.168.0.10** on port **631** (purple line). Remember, all this traffic is passing trough corporate firewall as legit traffic, even if the illustration perhaps shows otherwise.

9.6 Automating the task

By now we should have covered the basics on how to bypass corporate firewall in order to get to your office computer and network equipment. Now, ssh -R isn' t really practical, it consumes one terminal, and as soon as it shuts down there is no tunnel and no outside connectivity for that matter. The easiest thing to do is putting a cron jo b that will connect to remote server if the connection fails, office computer reboots etc.

First of all generate ssh keys[16], and add them to ssh-agent so that script won' t ask you for remote server' s password or local key phassphrase all the time.

Next we will add two extra parameters to our command **-N** and **-f** so that the connection goes into the background.

the command will look like:

```
ssh -N -f -R [bind_address:]port:host:hostport
```

next we need a shell script that will be triggered by the cron. For this example we will use the Real life example 2:

[16] https://toic.org/blog/2008/ssh-basics/

```
#!/bin/sh
COMMAND="ssh -N -f -R 89.xxx.xx.4:2210:localhost:22 bserver.outside.com"
pgrep -f -x "$COMMAND" > /dev/null 2>&1 || $COMMAND
```

Edit this code so it suits your needs, and save it in your home dir as **reverse_ssh_tunnel.sh**

Now we need to add a crontab entry which will trigger this script every 5 minutes:

```
crontab -e
```

and add:

```
*/5 * * * * /bin/sh /home/username/reverse_ssh_tunnel.sh
```

If you are connecting to different user name on remote server you can edit your commands so they look like:

```
ssh -R [bind_address]:port:host:host_port username@remote_host
```

Have fun and be safe!

CHAPTER 10

BOSH task

BOSH 를 통한 모든 작업은 task 라고 하는 단위로 관리 된다. 그리고 task 를 조회하는 작업도 하나의 task 이므로 아래와 같이 watch 명령으로 반복 조회를 할 경우 어느 순간 task number 가 순식간이 올라가 있는 것을 발견하게 될 것이다. 조심하자.

10.1 Tasks

```
   $ watch bosh -e lite tasks
   Using environment '192.168.50.4' as client 'admin'

   #    State   Started At                   Last Activity At             User      
↪Deployment  Description                  Result
   256  queued  Thu Jan  1 00:00:00 UTC 1970  Mon May 22 07:00:00 UTC 2017  scheduler  - 
↪         scheduled SnapshotDeployments  -
   255  queued  Thu Jan  1 00:00:00 UTC 1970  Mon May 22 06:53:46 UTC 2017  admin      cf-
↪mysql    delete deployment cf-mysql     -
   254  queued  Thu Jan  1 00:00:00 UTC 1970  Mon May 22 06:53:26 UTC 2017  admin      cf-
↪mysql    retrieve vm-stats              -
   253  queued  Thu Jan  1 00:00:00 UTC 1970  Mon May 22 06:52:38 UTC 2017  admin      cf
↪         retrieve vm-stats              -
   252  queued  Thu Jan  1 00:00:00 UTC 1970  Mon May 22 06:49:39 UTC 2017  admin      cf-
↪mysql    delete deployment cf-mysql     -
   251  queued  Thu Jan  1 00:00:00 UTC 1970  Mon May 22 06:49:14 UTC 2017  admin      cf
↪         retrieve vm-stats              -
   250  queued  Thu Jan  1 00:00:00 UTC 1970  Mon May 22 06:48:34 UTC 2017  admin      cf
↪         retrieve vm-stats              -
   249  queued  Thu Jan  1 00:00:00 UTC 1970  Mon May 22 06:48:16 UTC 2017  admin      cf-
↪mysql    delete deployment cf-mysql     -
   245  queued  Thu Jan  1 00:00:00 UTC 1970  Mon May 22 06:25:53 UTC 2017  admin      cf-
↪mysql    retrieve vm-stats              -
   244  queued  Thu Jan  1 00:00:00 UTC 1970  Mon May 22 06:24:50 UTC 2017  admin      cf-
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
↪mysql    retrieve vm-stats             -
   243  queued  Thu Jan  1 00:00:00 UTC 1970  Mon May 22 06:24:20 UTC 2017  admin      cf
↪          retrieve vm-stats             -
   242  queued  Thu Jan  1 00:00:00 UTC 1970  Mon May 22 06:22:43 UTC 2017  admin      cf
↪          retrieve vm-stats             -

   12 tasks

   Succeeded
```

10.2 cancel-task

```
   $ bosh -e lite cancel-task 256
```

이렇게 하여 진행중인 task 를 취소할 수 있다. 이것은 때때로 무한히 대기를 탈 수도 있다.