AI-DLC Initializer 완벽 가이드

AI-DLC 방법론 초기 셋팅 자동화 도구

---

1. AI-DLC Initializer란?

AI-DLC Initializer는 Raja SP(AWS)가 정의한 **AI 주도 개발 라이프사이클(AI-DLC)** 방법론을 프로젝트에 적용하기 위한 초기 셋팅을 자동화하는 도구입니다.

핵심 기능

- 프로젝트 정보 수집 및 분석
- 프로젝트 목적/목표 보고서 자동 생성
- AI-DLC 적용방식 보고서 자동 생성
- 프로젝트 맞춤형 AI-DLC 프롬프트 세트 생성

지원 프로젝트 상태

상태	설명
신규 프로젝트	빈 폴더에서 시작하는 새 프로젝트
리팩토링	기존 구현된 프로젝트 개선
기능 추가	기존 프로젝트에 새 기능 개발

---

2. AI-DLC 방법론 개요

배경

소프트웨어 엔지니어링의 진화는 개발자가 저수준의 차별화되지 않은 작업을 추상화하여 복잡한 문제 해결에 집중할 수 있도록 하는 지속적인 탐구였습니다. AI가 진화함에 따라 그 적용은 코드 생성을 넘어 요구사항 상세화, 계획, 작업 분해, 설계, 개발자와의 실시간 협업으로까지 확장되고 있습니다.

기존 소프트웨어 개발 방법론은 인간 주도의 장기 프로세스를 위해 설계되었으며, AI의 속도, 유연성, 고급 기능과 완전히 일치하지 않습니다. AI-DLC는 AI의 기능을 완전히 통합하도록 설계된 재고된 AI 네이티브 방법론입니다.

AI-DLC의 철학

AI-DLC는 도구가 아니라 **철학**입니다. 특정 환경이나 AI 서비스에 종속되지 않으며, 중요한 것은 AI와 인간이 협업하는 방식입니다.

---

3. AI-DLC 핵심 원칙 (10가지)

원칙 1: 후추가가 아닌 재고

기존 SDLC나 애자일 같은 방법론에 AI를 후추가하는 대신, 개발 방법론을 재고합니다. 전통적 방법론은 더 긴 반복 기간(월이나 주 단위)을 위해 구축되었으며, AI의 적절한 적용은 시간이나 일 단위로 측정되는 급속한 사이클로 이어집니다.

원칙 2: 대화 방향의 역전

인간이 AI와 대화를 시작하여 작업을 완료하는 것이 아니라, **AI가 인간과의 대화를 시작하고 지시**합니다. AI는 고수준 의도를 실행 가능한 작업으로 분해하고, 권장사항을 생성하며, 트레이드오프를 제안합니다. 인간은 **승인자**로 기능하며, 중요한 분기점에서 선택지를 검증, 선택, 확인합니다.

비유: Google Maps에서 인간은 목적지(의도)를 설정하고, 시스템은 단계별 지시를 제공합니다. 도중에 인간은 필요에 따라 감독하고 여정을 조정합니다.

원칙 3: 설계 기술을 핵심으로 통합

스크럼이나 칸반 같은 애자일 프레임워크는 설계 기술(예: 도메인 주도 설계)을 범위 밖으로 두었습니다. AI-DLC는 이를 필수 핵심으로 가집니다. 도메인 주도 설계(DDD), 행동 주도 개발(BDD), 테스트 주도 개발(TDD) 등 프로젝트에 적합한 방법론을 선택합니다.

원칙 4: AI 능력과 정렬

AI가 고수준 의도를 실행 가능한 코드로 자율적으로 변환하거나 인간 감독 없이 독립적으로 운영할 신뢰성이 아직 없다는 것을 인식합니다. AI-DLC는 인간 참여와 현재 AI의 능력 및 한계의 균형을 맞추는 **AI 주도 패러다임**을 채택합니다.

원칙 5: 복잡한 시스템 구축 지원

AI-DLC는 지속적인 기능 적응성, 고급 아키텍처 복잡성, 다수의 트레이드오프 관리를 요구하는 시스템 구축에 대응합니다. 트레이드오프 관리가 거의 필요하지 않은 더 단순한 시스템은 로우코드/노코드 접근방식에 적합합니다.

원칙 6: 인간 공생을 강화하는 요소 유지

인간 검증과 위험 완화에 필수적인 기존 방법론의 산출물과 터치포인트를 유지합니다. 예: 사용자 스토리, 위험 레지스터 등.

원칙 7: 친숙함을 통한 전환 촉진

새로운 방법론은 광범위한 훈련을 요구하지 않아야 합니다. 기존 실무자는 하루 만에 오리엔테이션을 받고 실습을 시작할 수 있어야 합니다. 스프린트(주 단위)를 **볼트(Bolt, 시간/일 단위)**로 재브랜딩하여 빠른 사이클을 강조합니다.

원칙 8: 효율성을 위한 책임 간소화

AI의 작업 분해 및 의사결정 능력을 활용함으로써 개발자는 인프라, 프론트엔드, 백엔드, DevOps, 보안 등의 전통적 전문화 사일로를 초월합니다. 그러나 제품 소유자와 개발자는 프레임워크에 필수적입니다.

원칙 9: 단계 최소화, 흐름 최대화

자동화와 책임 수렴을 통해 인수인계와 전환을 최소화하고 지속적인 반복 흐름을 가능하게 합니다. 중요한 결정 분기점에서의 인간 감독을 위한 최소한이지만 충분한 단계를 통합합니다.

원칙 10: 하드와이어된 독단적 SDLC 워크플로 없음

AI-DLC는 다른 개발 경로에 대해 독단적 워크플로를 규정하지 않습니다. AI가 주어진 경로의 의도에 기반하여 레벨 1 계획을 권장하고, 인간은 대화형 대화를 통해 검증하고 조정합니다.

---

4. AI-DLC 핵심 프레임워크

핵심 산출물

산출물	설명
의도 (Intent)	비즈니스 목표, 기능 또는 기술적 성과를 캡슐화한 고수준 목적 선언. AI 주도 분해의 출발점
유닛 (Unit)	의도에서 파생된 결속력 있는 자기완결형 작업 요소. 측정 가능한 가치를 제공하도록 설계됨
볼트 (Bolt)	최소 반복 단위. **시간 또는 일 단위**로 측정되는 구축-검증 사이클 (스프린트의 현대화)
도메인 설계	인프라 컴포넌트와 독립적으로 유닛의 핵심 비즈니스 로직을 모델링하는 산출물
논리 설계	도메인 설계를 확장하고 적절한 아키텍처 설계 패턴을 사용하여 비기능 요구사항을 충족
배포 유닛	패키지화된 실행 가능 코드, 구성, 인프라 컴포넌트를 포함하는 운영 산출물

AI-DLC 3단계

1. 구상 단계 (Inception)

의도를 포착하고 이를 개발을 위한 유닛으로 변환합니다.

모브 엘라보레이션(Mob Elaboration) 의식:

- 퍼실리테이터가 주도하는 공유 화면이 있는 단일 룸에서 수행

- AI가 의도를 사용자 스토리, 수용 기준, 유닛으로 초기 분해를 제안
- 제품 소유자, 개발자, QA, 기타 이해관계자가 협력적으로 검토하고 세련화

구상 단계 산출물:

- PRFAQ (선택)
- 사용자 스토리
- 비기능 요구사항(NFR) 정의
- 위험 설명
- 측정 지표
- 명확히 정의된 유닛

2. 구축 단계 (Construction)

정의된 유닛을 테스트된 운영 준비 배포 유닛으로 변환합니다.

진행 순서:

1. 도메인 설계 - 비즈니스 로직 모델링
2. 논리 설계 - NFR과 아키텍처 패턴 적용
3. 코드 생성 - 실행 가능 코드 생성
4. 테스트 - 기능, 보안, 성능 테스트

모브 구축(Mob Construction) 의식:

- 모든 팀이 단일 룸에 배치
- 통합 사양 교환, 결정, 볼트 제공

3. 운영 단계 (Operations)

시스템의 배포, 관찰 가능성, 유지보수에 초점을 맞춥니다.

주요 활동:

- 배포 유닛 패키지화 및 롤아웃
- 텔레메트리 데이터 분석 (메트릭, 로그, 트레이스)
- 이상 감지 및 SLA 위반 예측
- 인시던트 런북 통합 및 권장사항 제안

---

5. AI-DLC 확장 해석: 프로젝트 분석 기반 적용

핵심 철학의 유효성

원본 AI-DLC 백서가 제시하는 핵심 원칙들은 AI와 함께 복잡한 문제를 해결하는 **모든 프로젝트에 적용 가능**합니다.

핵심 철학	설명
대화 방향의 역전	AI가 계획을 제안하고 인간이 검증하는 방식
볼트 사이클	시간/일 단위의 빠른 반복으로 지속적인 검증과 피드백
계획-승인-실행 패턴	AI가 계획 수립 → 인간 검토/승인 → 실행

프로젝트 분석 기반 적용

AI-DLC는 특정 방법론(DDD 등)을 강제하지 않습니다. 프로젝트의 특성을 먼저 분석하고 그에 적합한 접근방식을 선택합니다.

분석 요소:

요소	고려 사항
프로젝트 상태	신규/개선/기능추가

프로젝트 성격	무엇을 만드는가, 최종 산출물은 무엇인가
도구와 환경	이미 선택된 것 또는 새로 선정 필요
복잡도	문제의 복잡도, 이해관계자, 제약사항, 의존성

중요: AI-DLC의 가치는 특정 기술이나 방법론에 있지 않습니다. AI와 인간이 협업하여 복잡한 문제를 해결하는 **방식을 정의**한 데 있습니다.

---

6. AI-DLC Initializer 실행 플로우

```
1. 스킬 실행 (/aidlc-init)
   ↓
2. 정보 수집 (AI 질문 → 사용자 답변)
   - 작업 주제
   - 프로젝트 상태 (신규/리팩토링/기능추가)
   - 프로젝트 설명
   - 프롬프트 패턴 선택 (태그 패턴 / Claude Code 패턴)
   ↓
3. [보고서 1] 프로젝트 목적/목표 보고서 생성
   → 사용자 검토 & 피드백
   ↓
4. [보고서 2] AI-DLC 적용방식 보고서 생성
   → 사용자 검토 & 피드백
   ↓
5. 폴더 생성 + AI-DLC 프롬프트 생성
   ↓
6. 준비 완료
```

---

7. 프롬프트 패턴

AI-DLC Initializer는 두 가지 프롬프트 패턴을 지원합니다.

태그 패턴 (범용)

모든 AI 도구(ChatGPT, Gemini, Claude 웹 등)에서 사용할 수 있는 범용 형식입니다.

특징:

- `[Question]` 과 `[Answer]` 태그로 질문과 응답 영역 구분
- 어떤 AI 도구에서도 복사하여 사용 가능
- 높은 이식성

형식 예시:

```
[Question]
프로젝트의 현재 상태는 무엇인가요?
- 신규 프로젝트
- 리팩토링
- 기능 추가
[/Question]

[Answer]
{사용자 응답}
[/Answer]
```

단점: 매번 복사-붙여넣기 필요, 선택형 질문 시 텍스트로 직접 선택

Claude Code 패턴 (Claude Code CLI 전용)

Claude Code CLI 환경에서만 사용 가능한 전용 형식입니다.

특징:

- `AskUserQuestion` 툴을 활용하여 터미널에서 직접 상호작용
- 선택형 UI 제공 (화살표 키로 옵션 선택)
- 여러 질문을 한 번에 묶어서 제시 가능
- 다중 선택(multiSelect) 옵션 지원

장점: 입력 오류 감소, 작업 속도 향상

단점: Claude Code CLI 환경에서만 동작, 다른 AI 도구로 이식 불가

---

8. 산출물 구조

AI-DLC Initializer 실행 후 생성되는 디렉토리 구조:

```
docs/aidlc-docs_{주제}/
├── project-goal-report.md              # 보고서 1: 프로젝트 목적/목표
├── application-approach-report.md      # 보고서 2: AI-DLC 적용방식
├── plan.md                             # 작업 계획 (체크박스)
├── prompts/                            # AI-DLC 프롬프트
│   ├── README.md                       # 프롬프트 사용 가이드
│   └── *.md                            # 프로젝트 특성에 맞게 생성
├── inception/                          # Inception 단계 산출물
├── construction/                       # Construction 단계 산출물
└── operations/                         # Operations 단계 산출물
```

---

9. AI-DLC 프롬프트 구조

모든 AI-DLC 프롬프트는 공통된 구조를 따릅니다:

1. 역할 정의

```
당신의 역할: 경험 많은 소프트웨어 엔지니어입니다.
```

AI에게 수행할 역할을 부여합니다.

2. 계획 수립 지시

```
아래 언급된 작업을 시작하기 전에 계획을 세우고
계획의 각 단계에 체크박스가 있는 md 파일에 단계를 작성하세요.
```

작업 전 계획을 세우고 파일에 기록하도록 합니다. 계획-승인-실행 패턴의 핵심입니다.

3. 승인 요청

```
계획을 생성하면 제 검토와 승인을 요청하세요.
제 승인 후에만 작업을 진행할 수 있습니다.
```

인간의 검토와 승인 없이 실행하지 않도록 명시합니다.

4. 작업 정의

```
당신의 작업: [구체적인 작업 내용]
```

수행해야 할 작업을 명확하게 설명합니다.

---

10. AI-DLC 실습 프롬프트 예시

설정 프롬프트

```
오늘은 애플리케이션 구축에 작업합니다. 모든 프론트엔드와 백엔드 컴포넌트를
위한 프로젝트 폴더를 생성하세요. 모든 문서는 aidlc-docs 폴더에 저장됩니다.
세션 전체에서 먼저 작업을 계획하고 그 계획을 위한 md 파일을 생성하도록 요청하세요.
제가 승인한 후에만 작업을 진행할 수 있습니다.

계획: aidlc-docs/plans 폴더
요구사항/기능 변경 문서: aidlc-docs/requirements 폴더
사용자 스토리: aidlc-docs/story-artifacts 폴더
아키텍처/설계 문서: aidlc-docs/design-artifacts 폴더
모든 지시사항: aidlc-docs/prompts.md 파일
```

구상 단계 - 사용자 스토리 생성

```
당신의 역할: 전문 제품 관리자로서 명확히 정의된 사용자 스토리를
생성하는 임무를 맡았습니다.

먼저 작업을 계획하고 각 단계에 체크박스가 있는 md 파일에 단계를 작성하세요.
단계에서 제 명확화가 필요하면 확인을 위해 단계에 메모를 추가하세요.
중요한 결정을 스스로 내리지 마세요.

계획을 완료하면 제 검토와 승인을 요청하세요.
제 승인 후 동일한 계획을 한 번에 한 단계씩 실행할 수 있습니다.

당신의 작업: [제품 설명]에 대한 사용자 스토리를 구축하세요.
```

구상 단계 - 유닛 분해

```
당신의 역할: 경험 많은 소프트웨어 아키텍트입니다.

작업을 시작하기 전에 계획을 세우고 units_plan.md 파일에 단계를 작성하세요.
중요한 결정을 스스로 내리지 마세요.

당신의 작업: 사용자 스토리 파일을 참조하여 독립적으로 구축할 수 있는
여러 유닛으로 그룹화하세요. 각 유닛에는 단일 팀이 구축할 수 있는
높은 응집도의 사용자 스토리가 포함됩니다.
유닛은 서로 느슨하게 결합됩니다.
```

구축 단계 - 도메인 모델 생성

```
당신의 역할: 경험 많은 소프트웨어 엔지니어입니다.

작업을 시작하기 전에 계획을 세우고 component_model.md 파일에 단계를 작성하세요.

당신의 작업: 사용자 스토리를 참조하여 모든 사용자 스토리를 구현하기 위한
컴포넌트 모델을 설계하세요. 이 모델에는 모든 컴포넌트, 속성, 행동,
```

```
컴포넌트가 사용자 스토리를 구현하기 위해 어떻게 상호작용하는지가 포함되어야 합니다.
아직 코드를 생성하지 마세요.
```

구축 단계 - 코드 생성

```
당신의 역할: 경험 많은 소프트웨어 엔지니어입니다.

작업을 시작하기 전에 계획을 세우고 md 파일에 단계를 작성하세요.

작업: 컴포넌트 설계 파일을 참조하여 설계에 있는 컴포넌트의
간단하고 직관적인 구현을 생성하세요.
깨끗하고 간단하며 설명 가능한 코딩 모범 사례를 따르세요.
```

---

11. 그린필드 vs 브라운필드

그린필드 개발 (신규 프로젝트)

1. **구상 단계**: AI가 명확화 질문 → 사용자 스토리/NFR/위험 설명 생성 → 유닛 그룹화 → PRFAQ 생성
2. **구축 단계**: 도메인 설계 → 논리 설계 → 코드 생성 → 테스트
3. **운영 단계**: 배포 → 관찰 가능성 및 모니터링

브라운필드 개발 (기존 프로젝트)

구축 단계에 추가 작업이 필요합니다:

1. AI가 기존 코드를 **더 높은 수준의 모델링 표현으로 승격**
   - 정적 모델: 컴포넌트, 설명, 책임, 관계
   - 동적 모델: 컴포넌트 상호작용 방식
2. 개발자가 역공학한 모델 검토/검증/수정
3. 이후 그린필드와 동일하게 진행

---

12. 모범 사례

DO (해야 할 것)

- 참조 백서를 먼저 읽고 이해한 후 실행
- 각 단계에서 사용자의 검토와 피드백 수신
- 보고서 생성 후 반드시 사용자 승인 후 다음 단계 진행
- 기존 프로젝트인 경우 설정 파일, 디렉토리 구조 분석 수행

DO NOT (하지 말아야 할 것)

- 사용자 승인 없이 다음 단계로 넘어가지 않음
- 적용방식 보고서에 코드 예시를 포함하지 않음
- 기존 폴더가 있을 때 확인 없이 덮어쓰지 않음
- 중요한 결정을 스스로 내리지 않음

---

13. AI-DLC 도입 방법

실습을 통한 학습

AI-DLC는 실제로 그룹으로 실습할 수 있는 일련의 의식(모브 엘라보레이션, 모브 구축 등)입니다. 문서나 전통적 훈련 대신, 실무자가 현재 해결하고 있는 실제 시나리오에서 AI-DLC 가이드와 함께 의식을 실습합니다.

개발자 경험 도구에 통합

SDLC를 가로질러 개발자에게 통합된 경험을 제공하는 오케스트레이션 도구에 AI-DLC를 통합하면, 개발자는 중요한 채택 드라이브 없이 원활하게 AI-DLC를 실습할 수 있습니다.

---

14. MCP 서비스화 가능성

AI-DLC Initializer는 MCP(Model Context Protocol) 서버로 변환하여 유료 API 서비스로 제공할 수 있습니다.

잠재적 MCP 도구

도구	설명
`aidlc_analyze_project`	프로젝트 분석 및 적합한 접근방식 권장
`aidlc_generate_goal_report`	프로젝트 목적/목표 보고서 생성
`aidlc_generate_approach_report`	AI-DLC 적용방식 보고서 생성
`aidlc_generate_prompts`	프로젝트 맞춤형 프롬프트 세트 생성
`aidlc_get_whitepaper`	AI-DLC 백서 내용 조회
`aidlc_get_templates`	프롬프트 템플릿 조회

비즈니스 모델 구상

- API 키 기반 인증
- 사용량 기반 과금 (크레딧 시스템)
- 구독 플랜 (Free/Basic/Pro/Enterprise)

---

15. 용어 정리

용어	설명
AI-DLC	AI 주도 개발 라이프사이클 (AI-Driven Development Lifecycle)
의도 (Intent)	달성해야 할 고수준 목적 선언
유닛 (Unit)	의도에서 파생된 자기완결형 작업 요소
볼트 (Bolt)	시간/일 단위의 최소 반복 사이클 (스프린트의 현대화)
모브 엘라보레이션	협력적 요구사항 상세화 및 분해 의식
모브 구축	협력적 구축 의식
DDD	도메인 주도 설계 (Domain-Driven Design)
BDD	행동 주도 개발 (Behavior-Driven Development)
TDD	테스트 주도 개발 (Test-Driven Development)
NFR	비기능 요구사항 (Non-Functional Requirements)
PRFAQ	Press Release / Frequently Asked Questions
ADR	아키텍처 결정 기록 (Architecture Decision Record)

이 문서는 Raja SP(AWS)의 AI-DLC 방법론과 AI-DLC Initializer 도구를 기반으로 작성되었습니다.