🧩

QA 자동화 테스트 결과 보고서_Team03

🧪 ECI Test Automation Project

1️⃣ 문서 개요 (Overview)

1.1 목적

> 본 문서는 **ECI 서비스**에 대해 수행한 **QA 자동화 테스트 결과를 기반**으로
> 자동화 범위, 테스트 실행 결과를 종합적으로 분석하고 서비스의 현재 품질과 주요 리스크를
> 판단하기 위해 작성되었습니다.
> 또한 동일한 테스트를 재현 가능한 형태로 정리함으로써 향후 **회귀 테스트 수행 및 품질 개선**을 위한
> 기준 자료로 활용하는 것을 목표로 합니다.

1.2 테스트 대상 및 범위

- **서비스명** : 📎☁️ECI 서비스
- **테스트 유형**
  - ☑️ **API 기능 테스트**
  - ☑️ **성능 테스트**
  - ☑️ **E2E 테스트**
- **테스트 범위**
  - **API 기능 테스트 범위**
    - 홈 및 인증 관련 API
    - 인프라에서의 리소스 관리 API
    - 컴퓨트 API
    - 네트워크 리소스 관리 API
    - 블록 스토리지 / 오브젝트 스토리지 API
    - 병렬 파일 시스템 API
  - **성능 테스트 범위**
    - 조회(GET) API 기반 부하·스파이크 테스트
    - 리소스 생성(POST) API 기반 장시간 안정성 검증(Soak Test)
  - **E2E 테스트 범위**
    - 가상 머신 생성 중심의 핵심 사용자 흐름
    - 리소스 CRUD 생명주기 검증(Create → Update → Delete)

2️⃣ 테스트 환경 (Test Environment)

항목	내용
OS	Windows 10
Language	Python 3.11.6
Test Framework	pytest
API Tool	pytest + requests
Performance Tool	Apache JMeter
CI	Jenkins
형상관리	GitLab
실행 방식	Jenkins CI 파이프라인 및 로컬 환경 실행

> 📌
> 본 테스트는 Windows 환경에서 실행 및 검증되었으며,
> 자동화 코드 레벨에서는 운영체제에 대한 의존성을 최소화하여
> 다양한 실행 환경으로의 확장이 가능하도록 구성하였습니다.

3️⃣ 전체 테스트 케이스 및 자동화 범위

3.1 테스트 케이스 분류 현황 (API, 성능, E2E)

- **API 테스트**

구분	테스트 케이스 수	비율
전체 테스트 케이스	158	100%
자동화 테스트	137	86.7%
수동 테스트	13	8.2%
미수행	8	5.1%

- **성능테스트**

구분	테스트 케이스 수	비율
전체 테스트 케이스	20	100%
자동화 테스트	20	100%
수동 테스트	0	0%
미수행	0	0%

- **E2E 테스트**

구분	테스트 케이스 수	비율
전체 시나리오	19	100%
자동화 시나리오	19	100%
수동 시나리오	0	0%
미수행	0	0%

3.2 자동화 범위 선정 기준

▪ API 자동화 기준

- API 응답이 명확하며, 결과를 객관적으로 검증 가능한 경우
- 반복 실행이 가능하고, 회귀 테스트 시 영향도가 높은 기능
- 데이터 의존성이 낮거나 테스트 데이터 제어가 가능한 시나리오
- 테스트 수행 시 자동화를 통해 실질적인 효율 향상이 기대되는 경우

구분	의미	API 예시	처리 방식
🟢자동화 가능(Yes)	•API 호출 및 응답을 통해 결과를 명확히 검증 가능 • 응답 값이 고정/예측 가능하며 반복 실행 가능	- 인증 토큰 발급 API - 리소스 생성 API - 상태 조회 API - 에러 코드/메시지 검증	즉시 자동화 대상
🟡조건부 가능(Later)	• API 자동화는 가능하나 비동기 처리·외부 의존·상태 전이 검증 난이도 높음	- 비동기 인스턴스 생성 API - 로그/메트릭 집계 API - 배치성 작업 API	1차 자동화 제외 → 안정화 후 고려
🔴 자동화 어려움(No)	• 응답만으로 성공/실패 판단 불가 사람의 해석·종합 판단 필요	- AI 분석 결과 API - 추천/랭킹 API - 통계 요약 API	수동 테스트 유지

▪ 성능 테스트 자동화 기준

- 사용자 기준 가장 많이 수행하는 행동
- 부하 패턴 및 검증 지표가 명확한 시나리오
- CI 환경에서 반복 실행 시 안정적으로 결과를 수집할 수 있는 케이스

▪ E2E 테스트 자동화 기준

- 흐름이 명확하고 실제 사용자의 흐름을 대표할 수 있는 시나리오
- 테스트 간 상태 공유 및 충돌 가능성이 통제 가능한 케이스
- 실패 시 원인 분석이 명확한 시나리오

▪ 자동화 제외 기준

- 리소스 생성/정리에 대한 의존성이 높은 케이스
- 환경별 데이터 편차가 큰 기능

3.3 API 테스트 기능별 자동화 커버리지

> 아래 표는 API 테스트를 기준으로 기능 영역별 테스트 케이스를 분류하고,
> 현재 자동화가 가능한 테스트 케이스의 비율을 기준으로 자동화 커버리지를 정리한 것입니다.

기능 영역	전체 TC	자동화 가능 TC	조건부 가능 TC	자동화 제외 TC	자동화율
계정	2	1	0	1	50%
홈	7	7	0	0	100%
인프라	8	8	0	0	100%
컴퓨트	23	22	0	1	95.7%
네트워크	31	26	0	5	83.9%
블록 스토리지	31	18	13	0	58.1%
오브젝트 스토리지	54	53	0	1	98.1
병렬 파일 시스템	2	2	0	0	100%
총합	158	137	13	8	86.7%

4️⃣ 자동화 테스트 실행 결과 요약

성능 테스트(부하 수준 기반 품질 판단)

부하 구간	주요 결과	판단 결과
Stable (800/30/10)	Avg / P95 / Error Rate 기준 충족	정상 운영 가능
Upper (1100/40/30)	일부 지표 SLA 미충족	한계 상한선
Stress (1300/40/30)	다수 요청 실패 및 SLA 초과	운영 불가

API 기능 테스트

항목	수치
총 자동화 테스트 수	137
Pass	137
Fail	0
성공률	100%

E2E 테스트

항목	수치
총 자동화 테스트 수	19
Pass	19
Fail	0
성공률	100%

5️⃣ 대표 API 기능 테스트 자동화 케이스

> 본 장에서는 전체 API 기능 테스트 중 **각 도메인을 대표할 수 있는 주요 자동화 테스트 케이스**를 선정하여, 테스트 목적과 검증 포인트를 중심으로 정리하였습니다.

5.1 API 도메인별 대표 성공 케이스

▪홈

항목	내용
테스트 ID	**HM_001**
사전 조건	유효한 인증 토큰 보유
테스트 스텝	1. `GET /api/user/organization/resource_usage` 요청 전송 2. 응답 상태 코드 및 Response Body 확인
기대 결과	- HTTP 상태코드 **200 반환** - Response Body에 `compute(vcpu, memory, instance_types)` 및 `storage(capacity)` 정보 포함
수행 결과	✅ **PASS**

▪인프라

항목	내용
테스트 ID	**INF_001**
사전 조건	유효한 인증 토큰 보유
테스트 스텝	1. `GET /api/user/region?count=50` 요청 전송 2. 응답 상태 코드 및 Response Body 확인
기대 결과	- HTTP 상태코드 **200 반환**- 리전 목록이 배열 형태로 반환
수행 결과	✅ **PASS**

▪컴퓨트

항목	내용
테스트 ID	**CT_011**
사전 조건	1개 이상의 가상 머신 존재
테스트 스텝	1. 실행 대상 가상 머신 ID 확인 2. `POST /user/resource/compute/virtual_machine_allocation` 요청 전송
기대 결과	- HTTP 상태코드 **200 반환**
수행 결과	✅ **PASS**

▪네트워크

항목	내용
테스트 ID	**NT_021**
사전 조건	-유효한 토큰 보유 -삭제 대상 가상 네트워크에 하나의 서브넷 연결
테스트 스텝	1.유효한 토큰 포함해 `DELETE /user/resource/compute/virtual_network/{id}` 요청 전송 2.응답 상태 코드 및 Response body 확인
기대 결과	-HTTP 상태코드 409 반환 -Response Body의 code 값이 "subnet_found" -삭제 막는 서브넷 ID 반환
수행 결과	✅ **PASS**

▪블록 스토리지

▪오브젝트 스토리지

항목	내용
테스트 ID	**BLK_015**
사전 조건	블록 스토리지 생성 완료 및 연결 대상 머신 ID 확보
테스트 스텝	1. 연결 대상 VM ID 확인2. 수정 대상 블록 스토리지 생성 3. `attached_machine_id` 값 수정 4. `PATCH /user/resource/storage/block_storage/{id}` 요청 전송 5. 응답값의 ID로 재조회6. `attached_machine_id` 변경 여부 확인
기대 결과	- HTTP 상태코드 **200 반환**- Response Body에 수정된 ID 반환
수행 결과	✅ **PASS**

항목	내용
테스트 ID	**OBJ_040**
사전 조건	- 유효한 인증 토큰 보유- 사전에 생성된 사용
테스트 스텝	1. `PATCH /api/user/resource/storage/object_storage/us` 요청 전송2. Request Body에 테스트 데이터
기대 결과	- HTTP 상태코드 **200 반환**- Response에 함 - 후속 GET 요청 시 수정된 name 확인- 수정 필드는 기존 값 유지
수행 결과	✅ **PASS**

▪병렬 파일 시스템

항목	내용
테스트 ID	**PFS_001**
사전 조건	유효한 인증 토큰 보유
테스트 스텝	1. `GET /user/resource/storage/parallel_file_system` 요청 전송 2. 응답 상태 코드 및 Response Body 확인
기대 결과	- HTTP 상태코드 **200 반환**- Response Body가 배열 형태로 반환 - 생성된 파일 시스템이 없을 경우 빈 배열 `[]` 반환
수행 결과	✅ **PASS**

> ✔️ status code, 필수 필드 존재 여부, 데이터 타입 검증 포함

6️⃣ 실패 테스트 케이스 상세 분석

❌ 실패 케이스 1

항목	내용
테스트 ID	test_sg_013_pip_view_edit_delete
기능	공인 IP 생성 및 관리
시나리오	공인 IP 생성 후, UI 화면에서 사용자가 본인의 공인 IP 자원을 식별
기대 결과	생성된 공인 IP가 UI 상에서 식별 가능한 정보와 함께 명확히 구분되어 표시됨
실제 결과	- UI 상에서는 공인 IP를 식별할 수 있는 정보가 제공되지 않아, API(GET) 조회를 통해서만 자원을 확인할 수 있음 - 사용자는 해당 공인 IP가 본인이 생성한 자원인지 여부를 UI 상에서 확인할 수 없음
결과	Failed

▪ 원인 분석

- **생성 시 식별 정보 부족** : 공인 IP 자원은 정상적으로 생성되었으나, UI 화면에서는 전체 리스트로 돌아가, 해당 자원을 식별할 수 있는 명확한 정보(식별자, 이름, 생성 시점 등)가 제공되지 않았습니다.
- 이로 인해 사용자는 목록 상에서 자신의 공인 IP를 직관적으로 구분할 수 없으며, 실질적으로는 API 조회를 통해서만 자원 식별이 가능한 상태였습니다.
- 이는 UI와 API 간 제공 정보의 불일치로 인해 자원 가시성(Resource Visibility)이 부족한 상태로 판단됩니다.

▪ 개선 방향

- **UI 사용자를 위한 식별자 안내 필요** : UI 상에서도 공인 IP 자원을 식별할 수 있도록 API에서 제공되는 식별자 또는 의미 있는 메타데이터를 함께 노출할 필요가 있습니다.
- 이를 통해 사용자가 별도의 API 호출 없이도 자원 상태를 직관적으로 확인하고 관리할 수 있도록 개선이 필요하다고 판단됩니다.

▪ 테스트 설계 변경

- 사용자 동작 흐름에 따른 기능 검증 범위를 보완하기 위해 API를 통해 생성된 공인 IP의 식별 정보를 기반으로 이후 UI 상의 사용자 동작을 검증하도록 테스트를 보완하였습니다.

❌ 실패 케이스 2

항목	내용
테스트 ID	test_sc_006_vm_create
기능	가상 머신 생성
시나리오	가상 머신 생성 시, 생성 필요한 하위 리소스 구성 흐름 및 UI 안내 확인
기대 결과	가상 머신 생성 시 네트워크 인터페이스(NIC)의 자동 생성 여부와 리소스 생성 흐름에 대한 명확한 안내 제공
실제 결과	가상 머신 생성 과정에서 NIC가 자동으로 생성되었으나, UI 상에서는 자동 생성 여부에 대한 안내가 제공되지 않아 사용자가 사전에 NIC를 생성해야 하는 것으로 오해할 수 있음
결과	Failed

▪ 원인 분석

- **명확한 안내 부족** : 가상 머신 생성 과정에서 네트워크 인터페이스는 별도의 사전 생성 없이 해당 단계에서 자동으로 생성되나, UI 상에서는 NIC의 자동 생성 여부에 대한 명확한 안내가 제공되지 않습니다.
- **생성 필요 리소스와의 혼동** : 간 생성 관계에 대한 직관적인 안내가 부족하며, 생성 페이지에서조차 다른 하위 리소스는 생성 필요성이 보이는 반면, NIC는 생성·선택 개념이 UI에서 명확히 구분되지 않아 사용자가 사전에 NIC를 생성해야 하는 것으로 오해할 수 있는 구조로 판단됩니다.

▪ 개선 방향

- **별도의 안내 필요** : 가상 머신 생성 단계에서 네트워크 인터페이스(NIC)의 자동 생성 여부와 다른 네트워크 리소스(가상 네트워크, 블록스토리지, 서브넷, Public IP)와의 생성·선택 관계를 명확히 안내할 수 있는 설명 또는 가이드 제공이 필요합니다.
- 이를 통해 사용자가 리소스 생성 흐름을 직관적으로 이해하고, 불필요한 사전 작업이나 혼선을 겪지 않도록 사용성 개선이 필요하다고 판단됩니다.

▪ 테스트 설계 변경

- 기존의 의존성을 지닌 하위 리소스들로 이루어진 스택을 생성할 때 네트워크 인터페이스를 생성하지 않는 방향으로 변경하였습니다.

7️⃣ 자동화를 통해 발견한 결함 요약

Bug ID	기능	내용	심각도	상태
3	활동 로그 조회	사용자 활동 로그 조회 시 타 계정의 활동 로그까지 노출됨	High	Open
4	로그인	로그인 완료 후 ECI 메인 화면으로 정상 이동되지 않거나, 시작하기 버튼이 간헐적으로 비활성화됨	High	Open
7	리소스 관리	각 도메인 리소스에 대해 타 계정 소유 리소스의 수정 및 삭제 권한 제한이 미흡함	High	Open

8️⃣ 성능 테스트 결과 분석

8.1 부하 기준 측정

▪ Stress 구간(1300 / 40 / 30)

- **측정 결과**
  - 평균 응답시간 3000 ms
  - P95 응답시간 6000ms
  - 에러율 2.95%(500error)
- **분석 및 판단**
  - 해당 부하 수준에서의 안정적인 서비스 운영이 어려운 것으로 판단했습니다.

▪ Upper 구간(1100 / 40 / 30)

- **측정 결과**
  - 평균 응답시간 2900 ms
  - P95 응답시간 6400ms
  - 에러율 0.02%(소수의 500error)
- **분석 및 판단**
  - 응답 시간 기준, 후반부 500에러를 통해 서비스 최대치로 판단했습니다.

▪ Stable 구간(800 / 30 / 10)

- **측정 결과**
  - 평균 응답시간 80 ms
  - P95 응답시간 200ms
  - 에러율 0.00%

- **분석 및 판단**
  - 안정적인 서비스 운영이 가능하다고 판단했습니다.
  - CI를 통한 회귀성능테스트를 실행할 수 있는 구간으로 선정했습니다.

8.2 평가 기준에 따른 분석

- **테스트 결과**
  - 개별 Load, Spike, Soak 모두 단독 수행 시 사전에 정의한 평가 기준 충족하며 통과했습니다.
  - **Spike Test**의 경우 이미 부하가 가해진 상태에서 연속적으로 적용한 경우 일부 응답 지표에서 기준 초과 관찰했습니다.
  - Load → Spike 이후 다시 수행한 **Load Test**에서는 전체 지표가 평가 기준을 충족하며 통과하는 것을 확인했습니다.
- **분석 결과**
  - **급격한 부하 증가** 시 일시적 성능 저하가 발생했습니다.
  - 단, 부하가 완화되면 자동적으로 정상 상태로 **회복 가능한 안정성**을 보유한 것으로 판단했습니다.

9️⃣ 자동화 테스트의 효과

항목	개선 효과
회귀 테스트 시간	수동 테스트 대비 테스트 수행 시간 단축으로 변경 사항 발생 시 빠른 재검증 가능
테스트 신뢰성	- Pytest 기반 일관된 검증 로직으로 테스트 결과의 재현성 확보 - 사람 실수로 인한 누락 최소화
성능 검증	부하 상황에서 SLA 충족 여부를 정량적 지표로 확인 가능
유지 보수성	테스트 코드 구조화로 기능 추가 및 수정 시 영향 범위 파악 용이
협업 효율성	공통 테스트 코드 및 실행 방식 통일로 팀원 간 환경 차이 최소화

🔟 향후 개선 계획

10.1 테스트 측면

- 검증 범위 확대
  - 정상 상황 외의 예외 및 Negative Test 케이스 추가
    - 인증 실패
    - 잘못된 입력값
    - 권한 오류
- soak 테스트 자동화 범위 확대
  - 장시간 실행에서의 안정성 검증을 위한 리소스 생성 검증
    - 단일 시나리오 중심의 진행 → 주요 리소스 생성 API 전반으로 확대

- CI 환경에서의 회귀 성능 테스트 단계 추가
  - 사이트 회복 후 Stable 구간 기준으로 지속적인 성능 검증 단계 추가
    - Jenkins에서 성능 저하 여부를 조기 탐지

10.2 기술적 개선

- 공통 API Client 및 Assert 로직 모듈화
  - API 기능 테스트의 유지보수성 ↑
  - 중복 코드 최소
- 테스트 실행 환경 간 일관성 개선
  - CI 환경과 로컬 환경 간의 테스트 결과 일관성 강화
    - 환경 변수 및 설정 관리 방식 정비
  - 로컬 간의 테스트 결과 일관성 강화
    - macOS 환경에서의 테스트 결과를 반영, 코드 분기 방식으로 정비

1️⃣1️⃣결론

> 본 QA 프로젝트에서는 **API 테스트, 성능 테스트, E2E 테스트**를 종합적으로 수행하여
> ECI 서비스의 주요 기능과 핵심 사용자 흐름에 대한 품질 안정성을 검증하였습니다.
>
> ✅**API 테스트**를 통한 각 기능 단위의 정상 동작 및 예외 상황을 반복적으로 검증,
> ✅**성능 테스트**를 통해 부하 상황에서도 SLA 기준 충족 여부를 정량적 지표로 확인,
> ✅ **E2E 테스트**를 실행하여 실제 사용자 관점에서의 주요 시나리오 흐름이 정상적으로 동작함을 검증하였습니다.
>
> 이를 통해 자동화 기반의 테스트 체계를 구축하고,
> **반복 테스트 효율 향상, 결함 조기 발견, 그리고 향후 기능 변경에 대한 신속한 품질 검증이 가능한 기반**을 마련하였습니다.