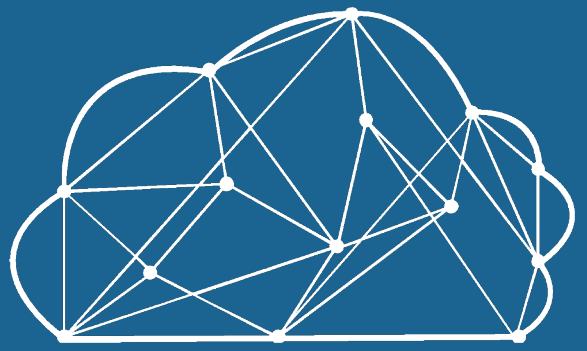
BESPIN GLOBAL



AWS Fargate 서비스와 nGrinder 융합으로 애플리케이션 부하 테스트 빠르게 시작하기

BESPIN GLOBAL
심선보 SEONBO SHIM
DevOps

스트레스 테스트 소개

애플리케이션의 스트레스 테스트는 애플리케이션이 특정 사용자 수나 트래픽을 처리할 수 있는 한계를 시뮬레이션하여, 시스템의 안정성과 성능을 평가하는 테스트로 이를 통해 시스템의 안정성, 확장성, 성능 등을 검증하고 개선할 수 있으므로 중요한 활동입니다.

1. 애플리케이션의 성능 검증
애플리케이션이 어떠한 상황에서도 일정한 성능을 유지할 수 있는지 확인합니다.

2. 애플리케이션의 안정성 검증
애플리케이션이 오류 또는 충돌을 일으키지 않고, 예상치 못한 상황에서도 안정적으로 동작할 수 있는지 확인합니다.

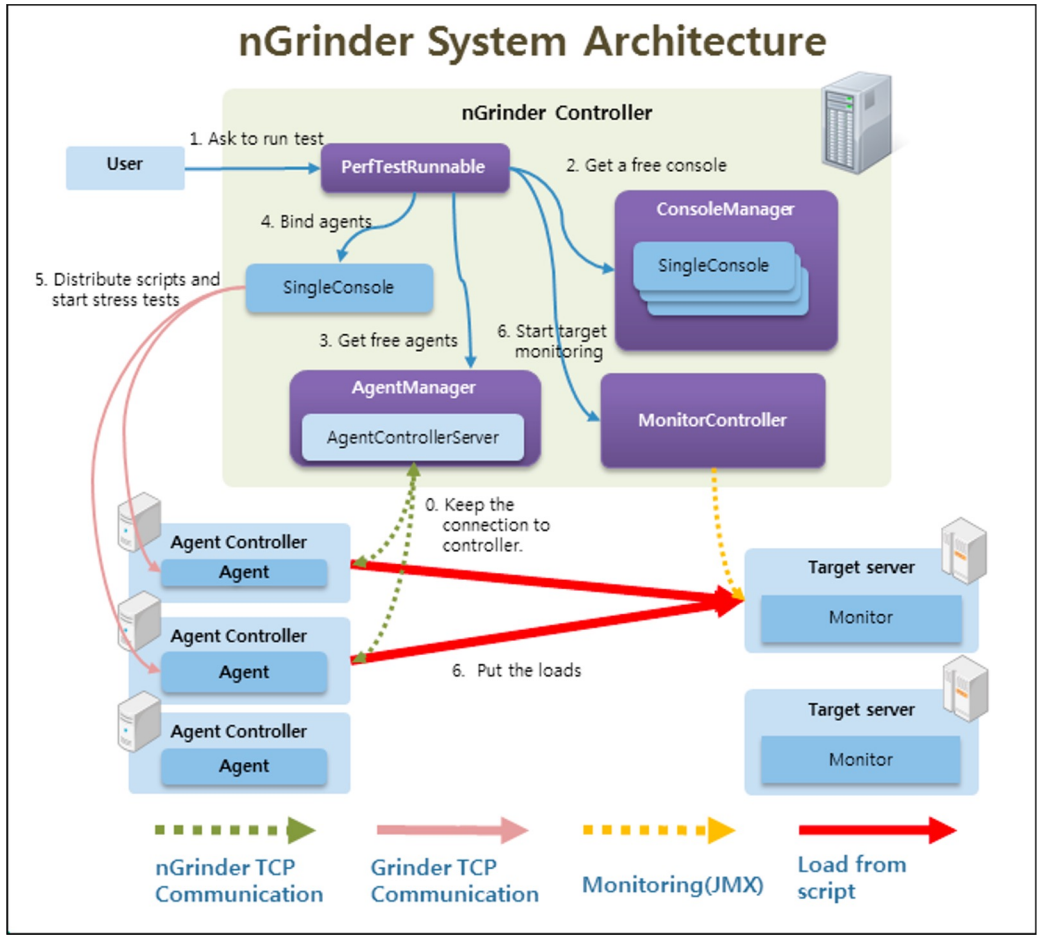
3. 확장성 검증
애플리케이션이 트래픽이 증가할 때 쉽게 확장될 수 있는지 확인합니다.

4. 문제점 및 병목 현상 발견
스트레스 테스트를 수행함으로써 애플리케이션의 문제점과 병목 현상을 식별할 수 있습니다.

5. 서비스 수준 계약(SLA) 준수 확인
애플리케이션이 SLA 항목을 계획하고 메트릭을 정의함으로써 고객에게 서비스 수준에 대한 보장을 제공할 수 있습니다.

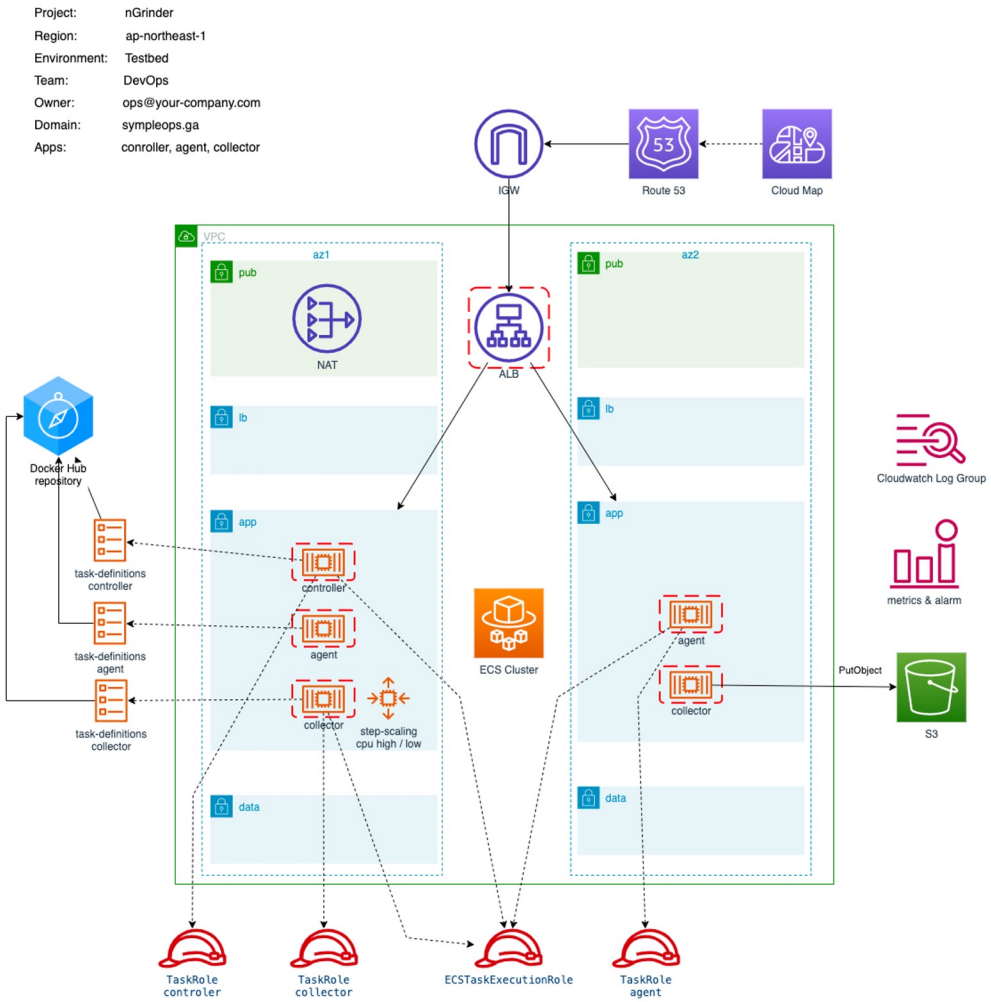
nGrinder 소개 및 아키텍처

nGrinder 는 Controller 와 Agent 로 구성된 클러스터 환경으로 동적으로 부하 규모를 구성 할 수 있을 뿐만 아니라, 쉽고 빠르게 테스트를 진행하고 진행 현황과 메트릭을 종합적으로 확인 할 수 있습니다.



Solutions Architecture for nGrinder Stack

nGrinder 부하 테스트를 위한 AWS 스택을 위한 솔루션즈 아키텍처로 다음과 같이 ECS 를 활용한 Serverless 기반으로 디자인 하였습

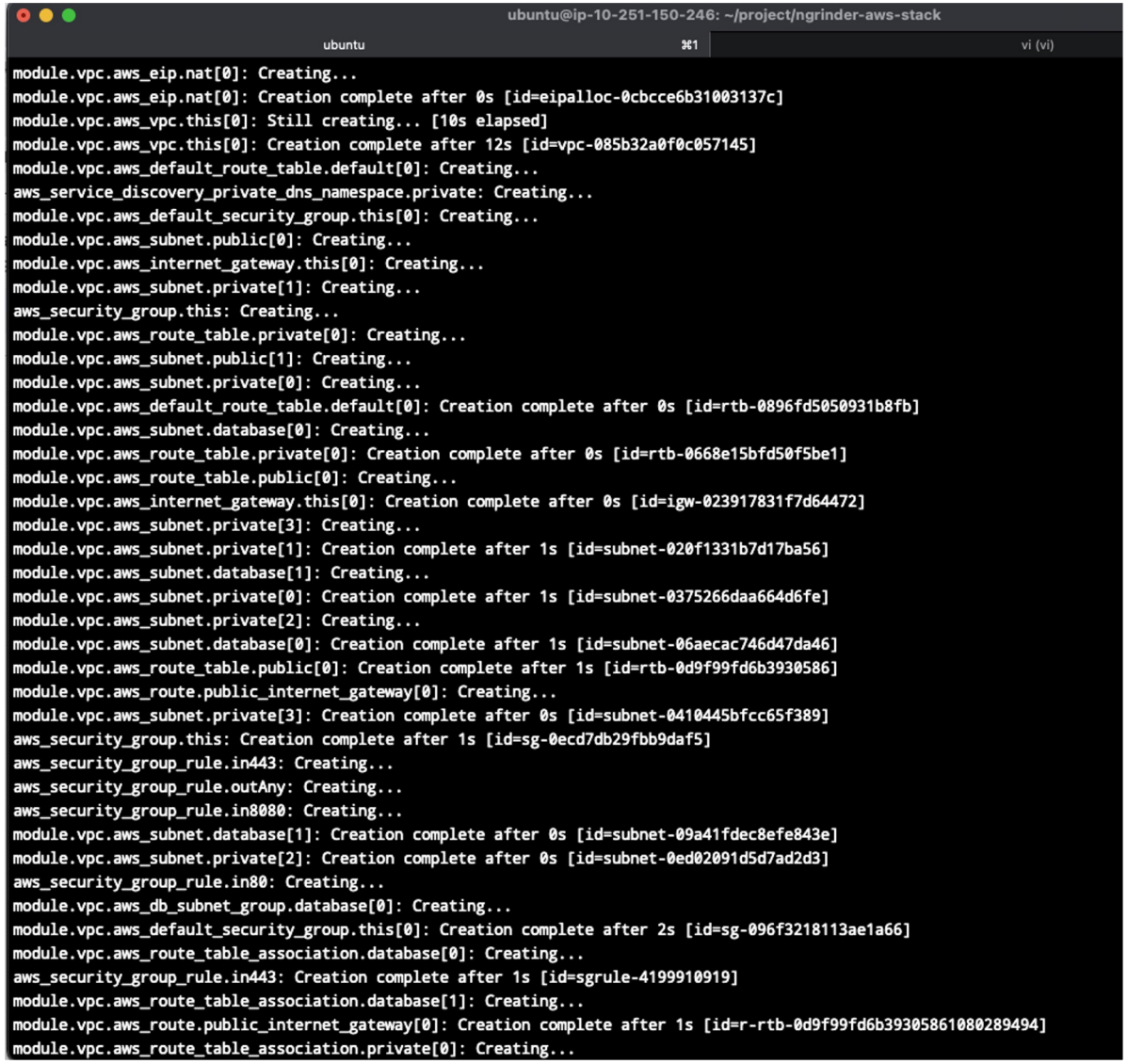


Name	Description
Project	ngrinder
Region	ap-northeast-1
Environment	Testbed
Team	DevOps
Domain	sympleops.ga
Private Domain	sympleops.local
Owner	admin@symplesims.io
Apps	controller, agent, collector

AWS 클라우드 설계시 일관된 리소스 관리를 위한 정책을 계획하기 위해 주요 Context 정보를 정의 하고

Terraform / CloudFormation 과 같은 프로비저닝 툴을 통해 각 리소스 및 태깅 속성을 자동화 구성할 수 있습니다.

nGrinder AWS Stack



nGrinder AWS Stack 을 프로비저닝 합니다.

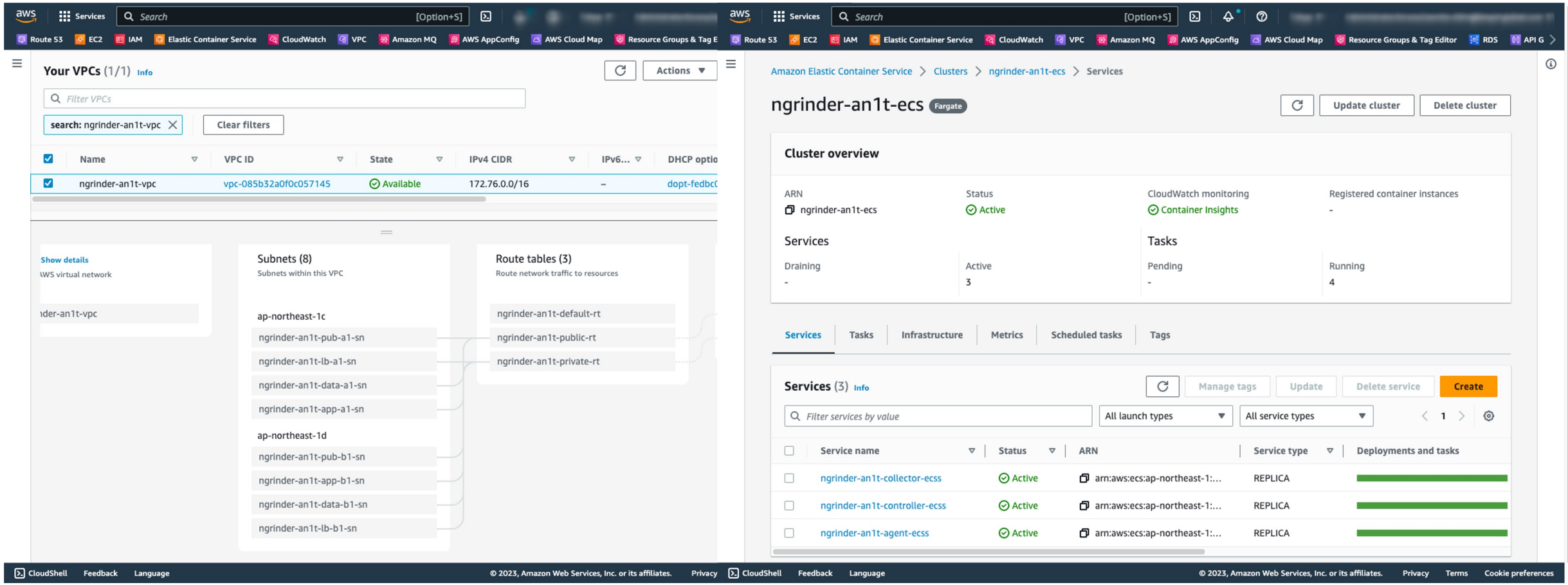
수 분 이내에 스트레스 테스를 할 수 있는 nGrinder 클러스터 환경이 AWS Fargate 서비스로 자동화 구성 됩니다.

놀랍게도 7분 정도 소요 되네요.....

It’s Demo Time

AWS 주요 구성 리소스 살펴 보기

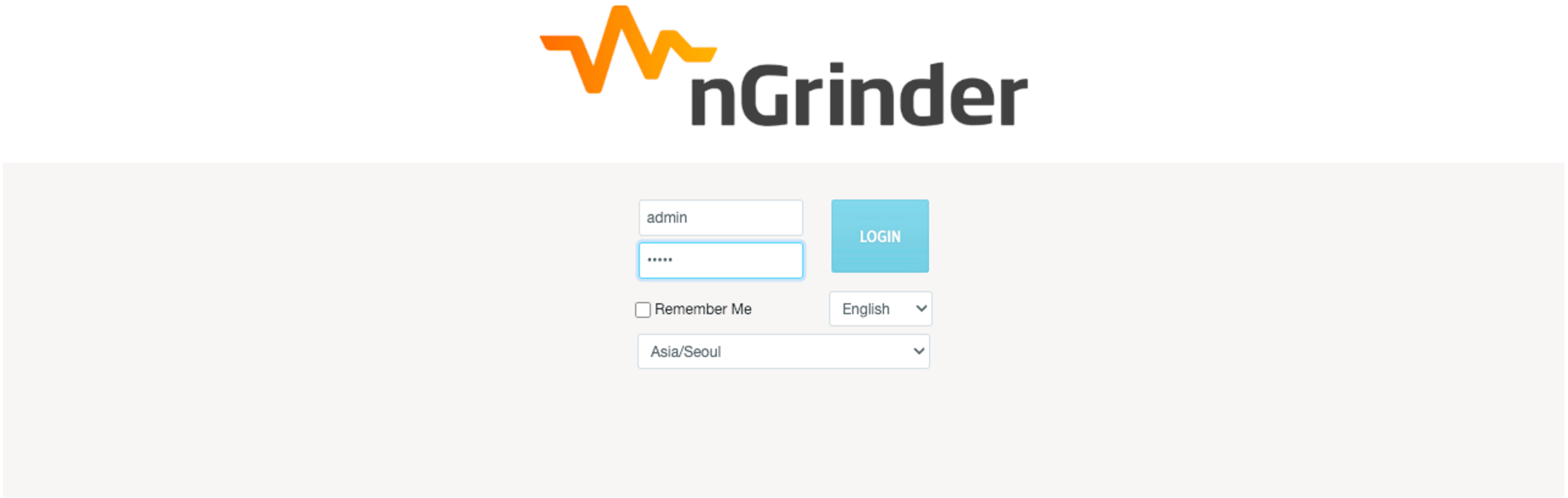
VPC / ALB / SG / ECS 클러스터 및 애플리케이션 서비스는 물론 Metric Alarm 과 Auto-Scaling 정책 적용까지 한번에 구성 됩니다.



AWS 서비스 뿐만 아니라 nGrinder 애플리케이션과 데이터 수집용 collector 애플리케이션 까지 구성 되었네요.

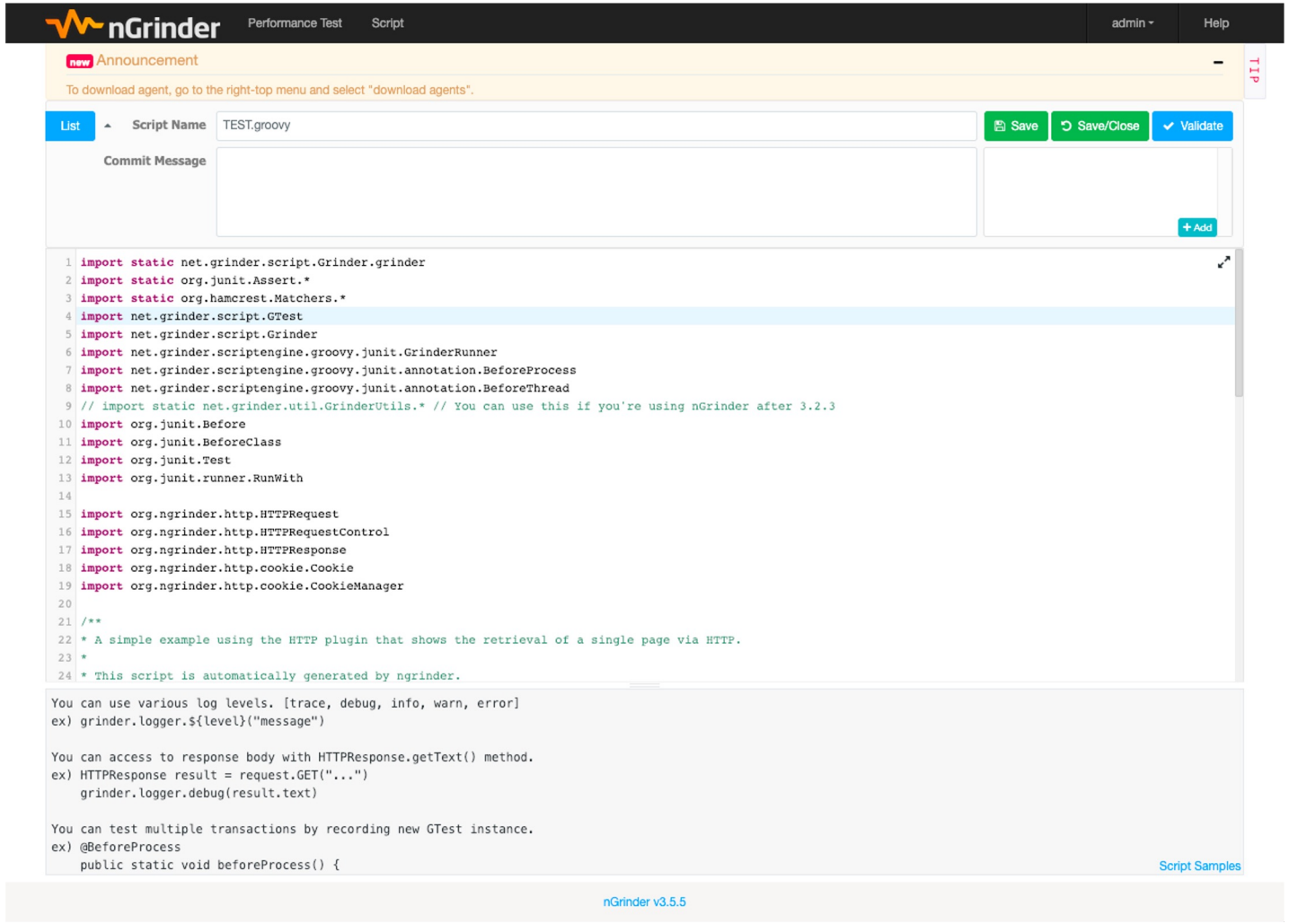
nGrinder 로그인

ALB 의 DNS 주소를 브라우저 주소창에 입력 하면 nGrinder 로그인 화면이 열립니다. 임시 인증은 admin / admin 이네요.



nGrinder 애플리케이션 부하 테스트 스크립트 작성

갑자기 코드가 나오니 당황하셨나요? 개발자들에게 너무나 익숙한 JUnit 기반으로 테스트 시나리오를 작성 할 수 있습니다. ^^.

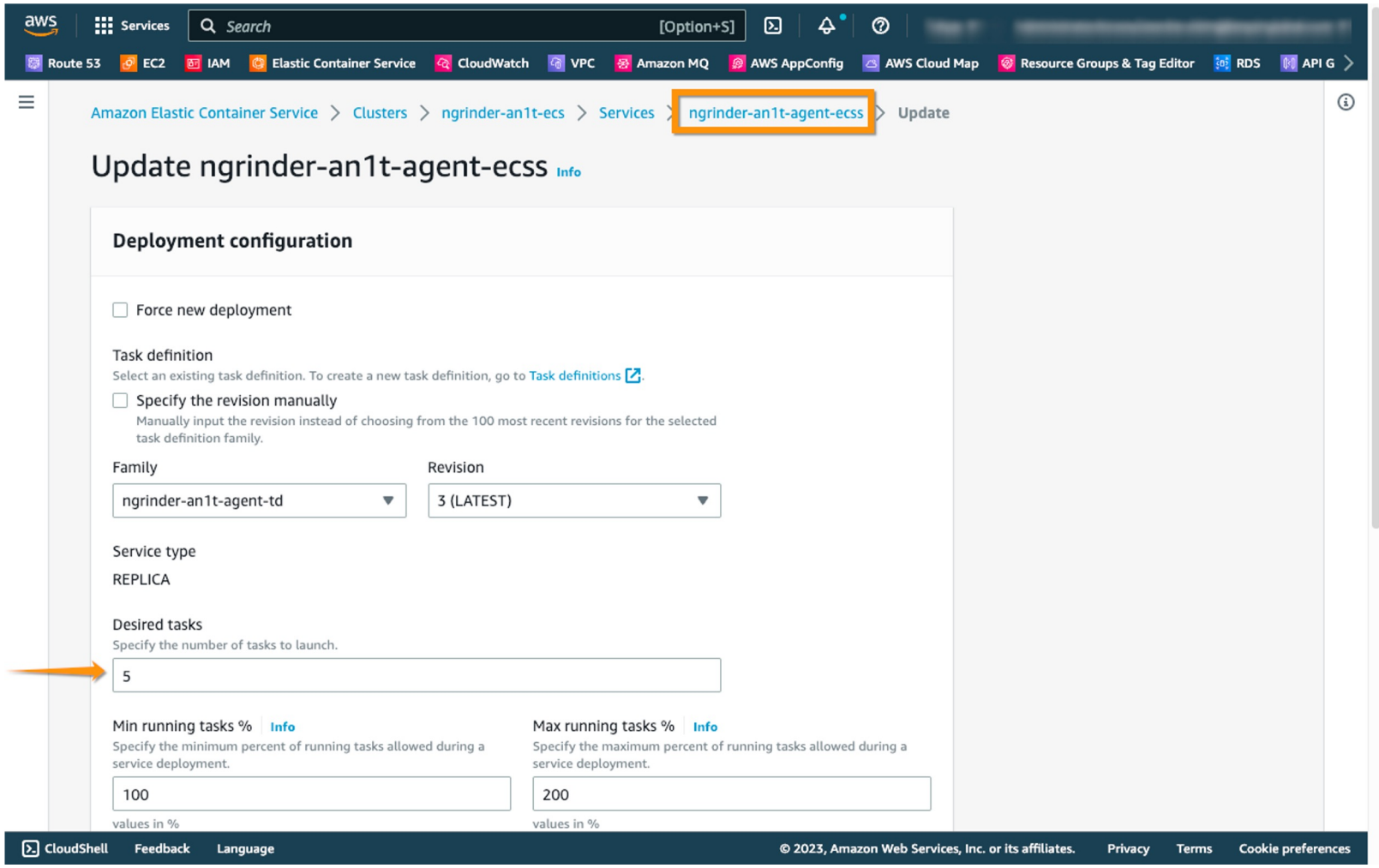


@Test 가 요청 입니다. Payload 만 동적으로 바꿔 보자구요.

```
static final Random rand = new Random()

String payload() {
    Integer id     = rand.nextInt(60001 - 10001) + 10001
    String name    = Long.toString(Math.abs(rand.nextLong() % 3656158440062976L),
36);
    Integer height = rand.nextInt(190 - 165) + 165
    Integer weight = rand.nextInt(100 - 46) + 46
    Integer year   = rand.nextInt(2003 - 1970) + 1970
    Integer mon    = rand.nextInt(12 - 1) + 1
    Integer day    = rand.nextInt(31 - 1) + 1
    Long    now    = System.currentTimeMillis()
    return "{ \"id\": \"I" + id + "\", \"name\": \"" + name + "\", \"birthday\": \"" + year + "-" +
mon + "-" + day  + "\", \"height\": "+height+", \"weight\": "+weight+",\n \"timestamp\":
"+now+"}"
}

@Test
public void test() {
  HTTPResponse response = request.POST("http://ngrinder-an1t-pub-alb-
993527877.ap-northeast-1.elb.amazonaws.com/api/collect", payload().getBytes())

  if (response.statusCode == 301 || response.statusCode == 302) {
    grinder.logger.warn("Warning. The response may not be correct. The response
code was {}.", response.statusCode)
  } else {
    assertThat(response.statusCode, is(201))
  }
}
```

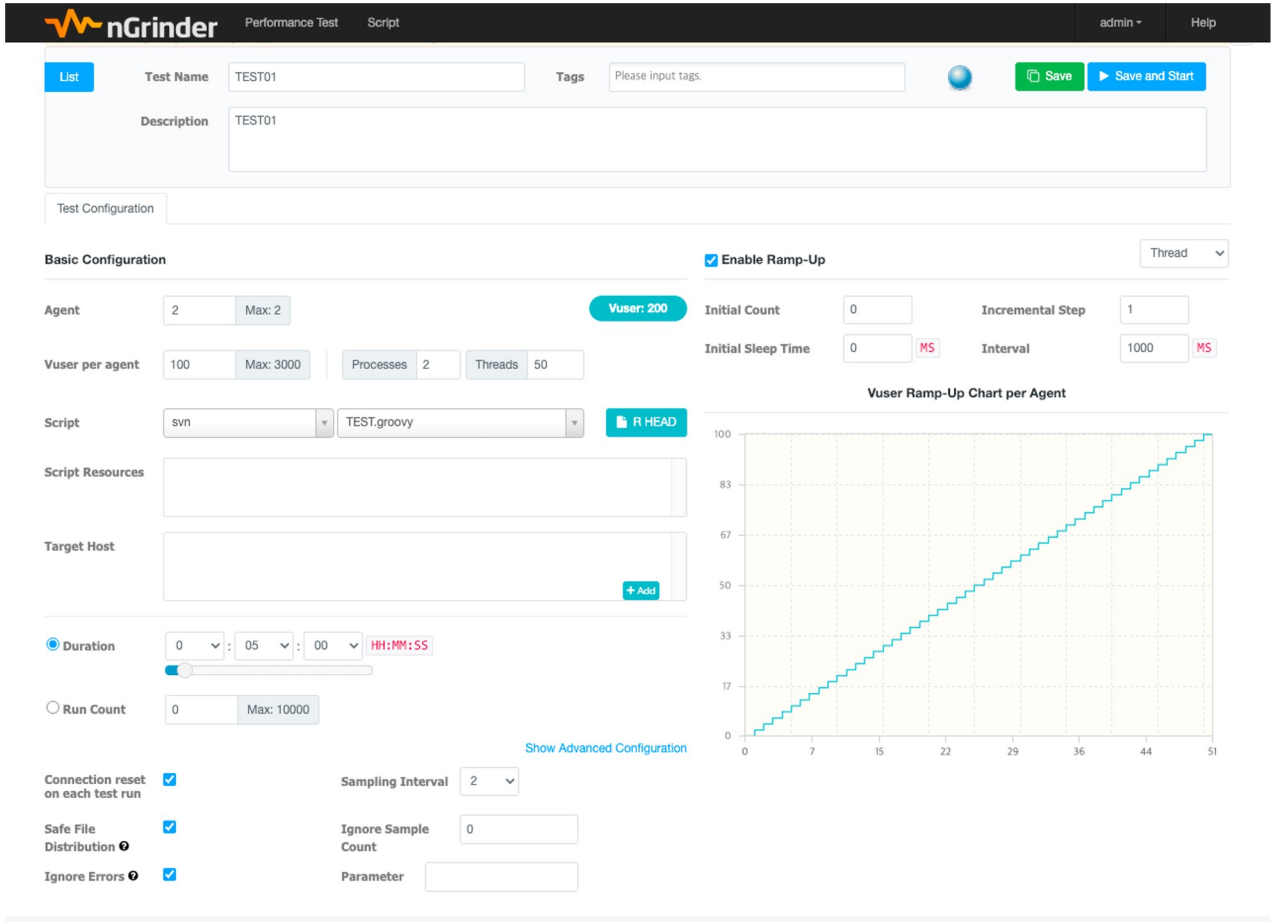
nGrinder 애플리케이션 부하 규모 설정

앤터프라이즈용 대규모 부하를 발생 하고 싶다면, agent 갯수를 원하는 만큼 조정 하세요.



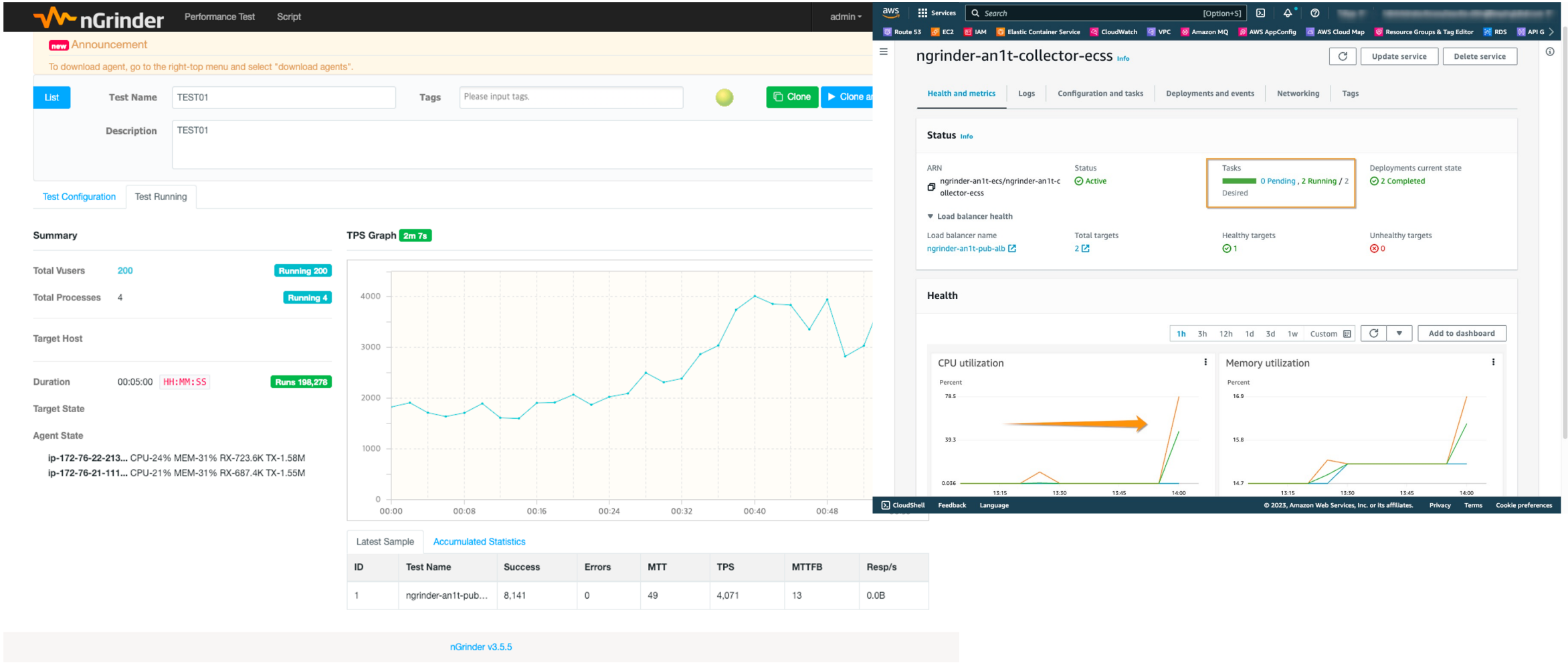
nGrinder 테스트 시나리오 계획

가상의 사용자 수(부하량)을 조정 하고 Ramp-Up 기능으로 성능 테스트에서 초기 사용자/요청 수를 점진적으로 증가시킬 수도 있습니다. 요청수 또는 부하 발생 기간을 지정하여 부하 테스트를 할 수 있습니다.



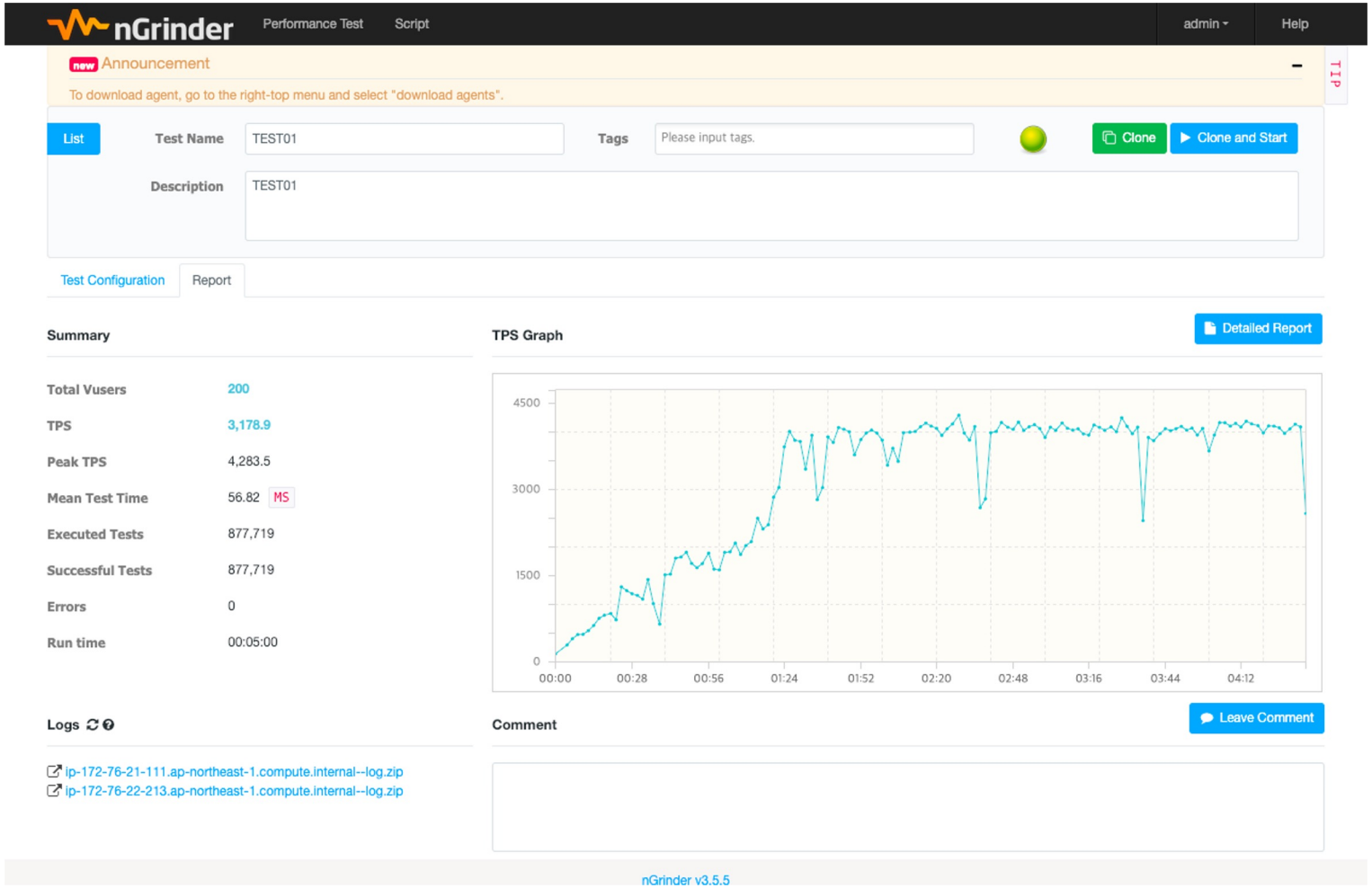
nGrinder 테스트 진행 및 모니터링

부하 진행 상황에 대응하는 애플리케이션의 주요 메트릭을 확인 할 수 있브니다. CPU 가 올라간다구요? ASG 가 트리거 될 것입니다. ^^



nGrinder 테스트 종료 및 리포팅

테스트가 종료되면 애플리케이션 성능 관련 주요 메트릭을 확인 할 수 있습니다. 테스트가 종료되면 애플리케이션이 scale-in 되겠죠 ^^, 덧붙여 S3 에 적재된 데이터도 살펴 볼수 있습니다.



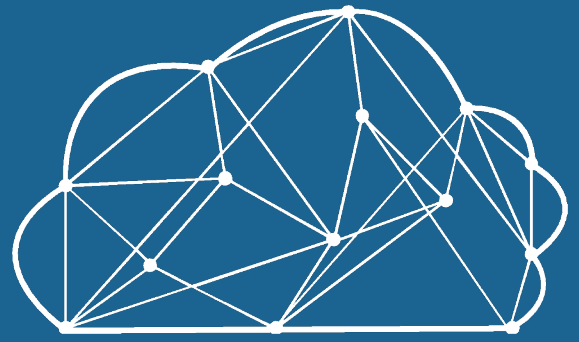
Conclusion

스트레스 테스트를 제대로 계획하고 이행하려면 레거시 환경에서는 매우 복잡하고 부하 테스트와 관련된 숙련된 전문가와 시간 및 비용이 많이 소요되었습니다.

여기선 AWS Fargate 와 nGrinder 오픈소스를 융합하여 짧은 시간에 테스트 환경을 위한 스택을 구성함은 물론, 개발자에 익숙한 스크립트로 테스트 시나리오를 작성하고, 작은 규모 에서부터 대규모 엔터프라이즈 환경까지 대응 하는 부하를 원하는 시간에 원하는 만큼 발생 할 수 있었으며, 부하에 반응하는 탄력적인 확장 능력을 갖춘 애플리케이션 또한 확인 할 수 있었습니다.

전통적인 On-Premise 환경에선 DevOps 스럽지 않았는데요, Cloud 환경에선 DevOps 답게 진화하였습니다.

Good Job!



감사합니다.

BESPIN GLOBAL