



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

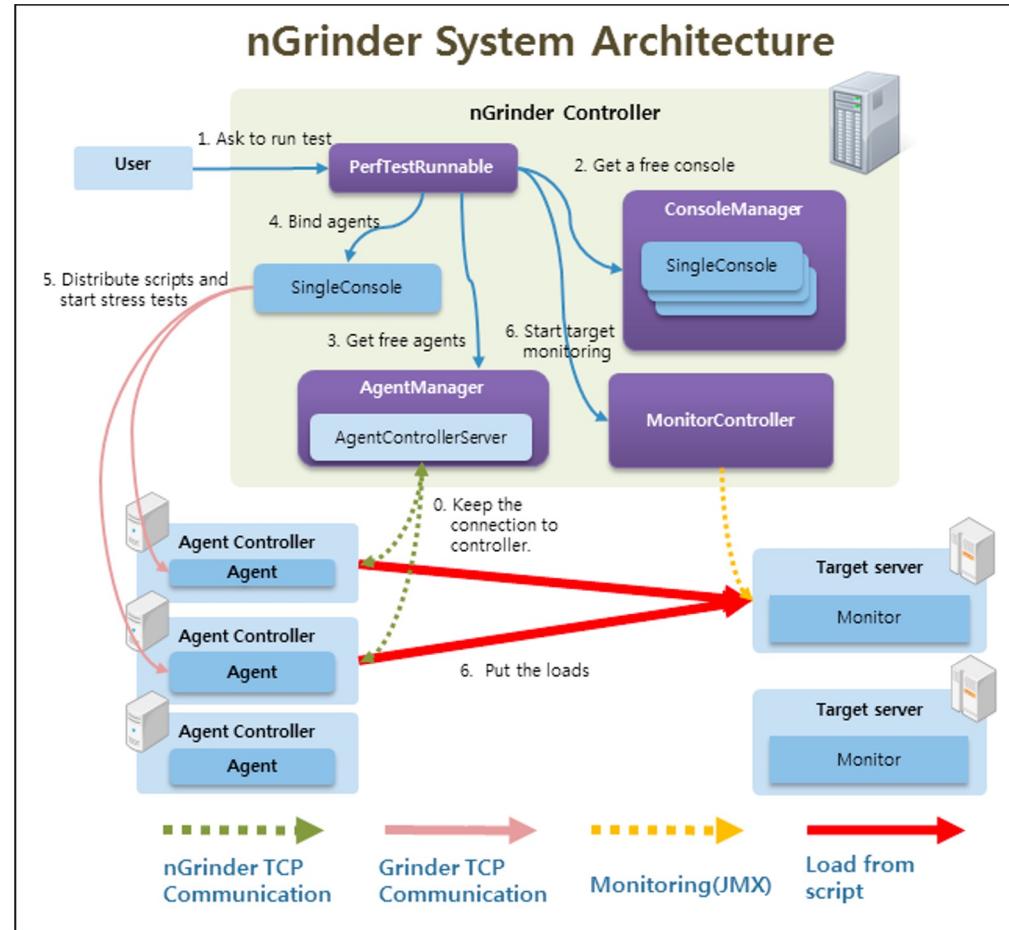
스트레스 테스트를 수행함으로써 애플리케이션의 문제점과 병목 현상을 식별할 수 있습니다.

## 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

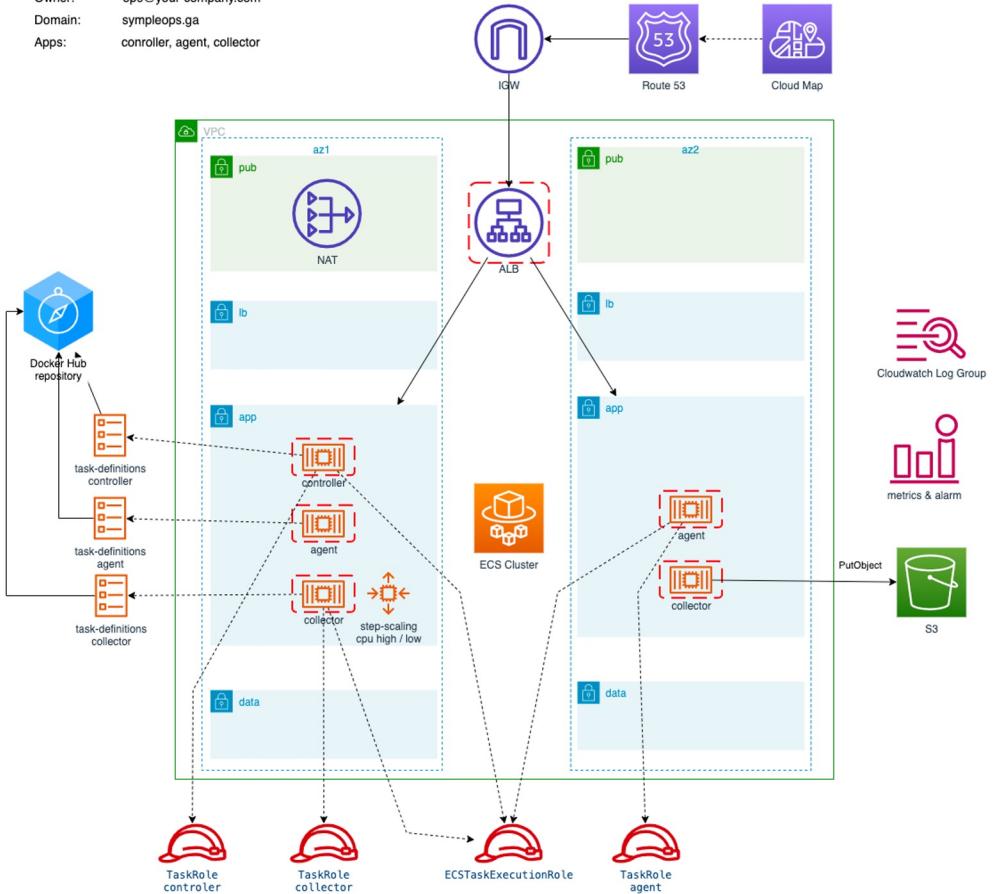
nGrinder는 Controller 와 Agent 로 구성된 클러스터 환경으로 동적으로 부하 규모를 구성 할 수 있을 뿐만 아니라, 쉽고 빠르게 테스트를 진행하고 진행 현황과 메트릭을 종합적으로 확인 할 수 있습니다.



# Solutions Architecture for nGrinder Stack

nGrinder 부하 테스트를 위한 AWS 스택을 위한 솔루션 아키텍처로 다음과 같이 ECS를 활용한 Serverless 기반으로 디자인 하였습니다.

Project:	nGrinder
Region:	ap-northeast-1
Environment:	Testbed
Team:	DevOps
Owner:	ops@your-company.com
Domain:	sympleops.ga
Apps:	controller, agent, collector



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

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

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

# nGrinder AWS Stack

```
ubuntu@ip-10-251-150-246: ~/project/ngrinder-aws-stack
ubuntu          *1          vi (vi)

module.vpc.aws_eip.nat[0]: Creating...
module.vpc.aws_eip.nat[0]: Creation complete after 0s [id=eipalloc-0cbcce6b31003137c]
module.vpc.aws_vpc.this[0]: Still creating... [10s elapsed]
module.vpc.aws_vpc.this[0]: Creation complete after 12s [id=vpc-085b32a0f0c057145]
module.vpc.aws_default_route_table.default[0]: Creating...
aws_service_discovery_private_dns_namespace.private: Creating...
module.vpc.aws_default_security_group.this[0]: Creating...
module.vpc.aws_subnet.public[0]: Creating...
module.vpc.aws_internet_gateway.this[0]: Creating...
module.vpc.aws_subnet.private[1]: Creating...
aws_security_group.this: Creating...
module.vpc.aws_route_table.private[0]: Creating...
module.vpc.aws_subnet.public[1]: Creating...
module.vpc.aws_subnet.private[0]: Creating...
module.vpc.aws_default_route_table.default[0]: Creation complete after 0s [id=rtb-0896fd5050931b8fb]
module.vpc.aws_subnet.database[0]: Creating...
module.vpc.aws_route_table.private[0]: Creation complete after 0s [id=rtb-0668e15bfd50f5be1]
module.vpc.aws_route_table.public[0]: Creating...
module.vpc.aws_internet_gateway.this[0]: Creation complete after 0s [id=igw-023917831f7d64472]
module.vpc.aws_subnet.private[3]: Creating...
module.vpc.aws_subnet.private[1]: Creation complete after 1s [id=subnet-020f1331b7d17ba56]
module.vpc.aws_subnet.database[1]: Creating...
module.vpc.aws_subnet.private[0]: Creation complete after 1s [id=subnet-0375266daa664d6fe]
module.vpc.aws_subnet.private[2]: Creating...
module.vpc.aws_subnet.database[0]: Creation complete after 1s [id=subnet-06aecac746d47da46]
module.vpc.aws_route_table.public[0]: Creation complete after 1s [id=rtb-0d9f99fd6b3930586]
module.vpc.aws_route_public_internet_gateway[0]: Creating...
module.vpc.aws_subnet.private[3]: Creation complete after 0s [id=subnet-0410445bfcc65f389]
aws_security_group.this: Creation complete after 1s [id=sg-0ecd7db29fbb9daf5]
aws_security_group_rule.in443: Creating...
aws_security_group_rule.outAny: Creating...
aws_security_group_rule.in8080: Creating...
module.vpc.aws_subnet.database[1]: Creation complete after 0s [id=subnet-09a41fdec8efe843e]
module.vpc.aws_subnet.private[2]: Creation complete after 0s [id=subnet-0ed02091d5d7ad2d3]
aws_security_group_rule.in80: Creating...
module.vpc.aws_db_subnet_group.database[0]: Creating...
module.vpc.aws_default_security_group.this[0]: Creation complete after 2s [id=sg-096f3218113ae1a66]
module.vpc.aws_route_table_association.database[0]: Creating...
aws_security_group_rule.in443: Creation complete after 1s [id=sgrule-4199910919]
module.vpc.aws_route_table_association.database[1]: Creating...
module.vpc.aws_route_public_internet_gateway[0]: Creation complete after 1s [id=r-rtb-0d9f99fd6b39305861080289494]
module.vpc.aws_route_table_association.private[0]: Creating...
```

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

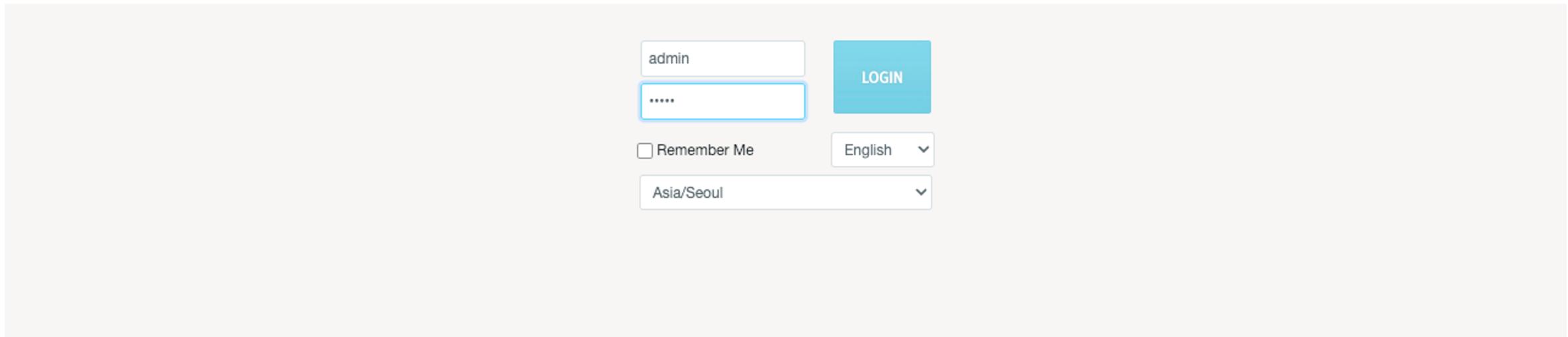
VPC / ALB / SG / ECS 클러스터 및 애플리케이션 서비스는 물론 Metric Alarm 과 Auto-Scaling 정책 적용까지 한번에 구성 됩니다.

The image displays two side-by-side screenshots of the AWS Management Console. The left screenshot shows the 'Your VPCs' page with one VPC named 'ngrinder-an1t-vpc'. It lists subnets (8) and route tables (3). The right screenshot shows the 'Cluster overview' page for an ECS cluster named 'ngrinder-an1t-ecs' (Fargate). It displays services (3), tasks (4), and container instances (4). Both screenshots show detailed configurations for each resource, including ARNs, statuses, and deployment metrics.

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

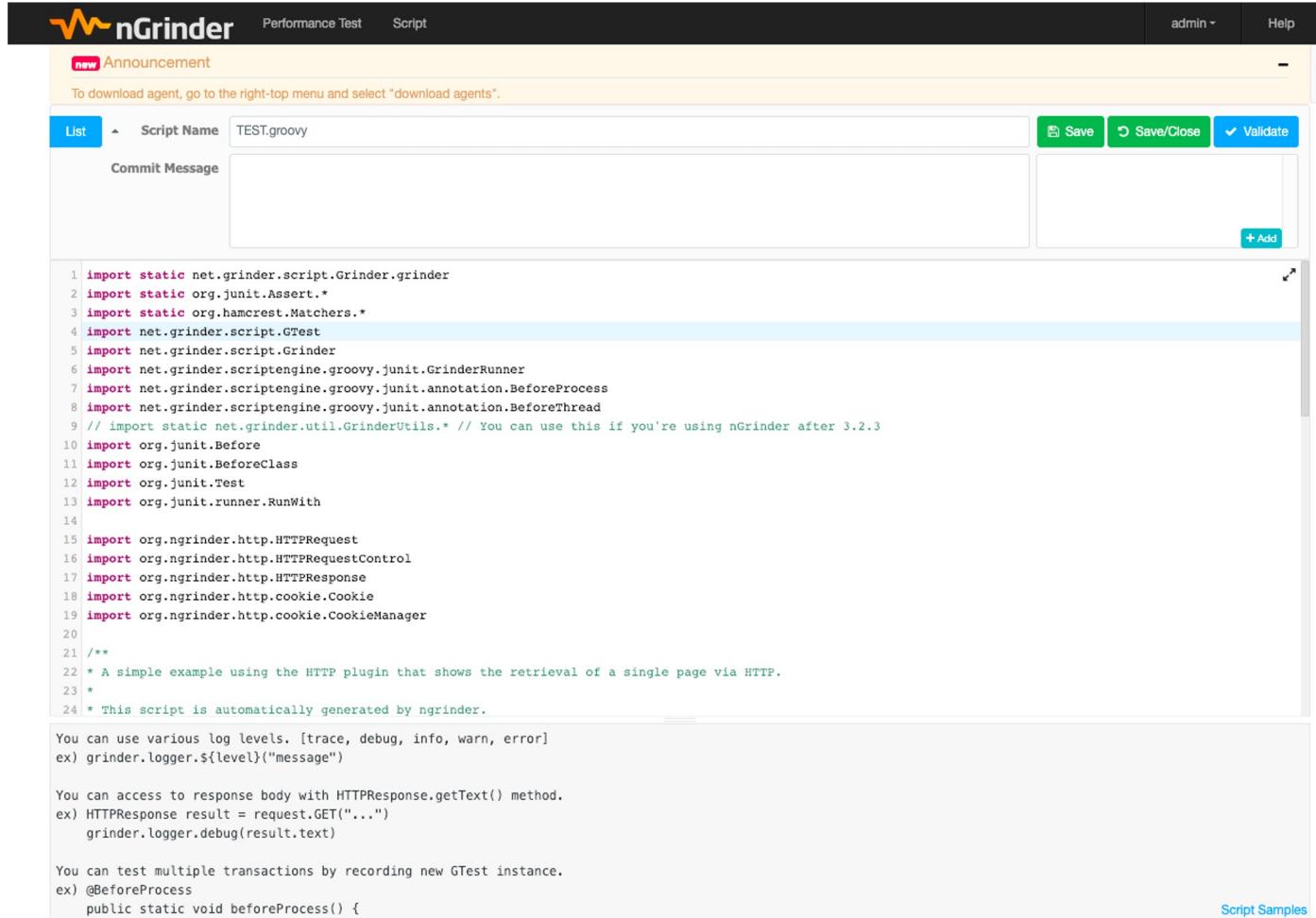
# nGrinder 로그인

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

A screenshot of the nGrinder login interface. It features a light gray header bar with the nGrinder logo. Below it is a white login form. The form includes two input fields: the first is pre-filled with "admin" and the second is empty with a blue outline, indicating it is the password field. To the right of these fields is a large blue "LOGIN" button. Below the password field is a checkbox labeled "Remember Me". To the right of the checkbox is a language selection dropdown set to "English". At the bottom of the form is another dropdown menu set to "Asia/Seoul".

# nGrinder 애플리케이션 부하 테스트 스크립트 작성

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



The screenshot shows the nGrinder web application interface. At the top, there's a navigation bar with the nGrinder logo, 'Performance Test', 'Script', 'admin', and 'Help'. Below the navigation is a banner with 'Announcement' and a link to download agents. The main area is a script editor titled 'TEST.groovy'. It contains Groovy code for a JUnit test. The code imports various nGrinder and JUnit classes, defines a random number generator, creates a payload object with random user data, and defines a test method that sends a POST request to a specific URL with the payload. Below the code editor, there are sections for 'Commit Message' and 'Script Samples'.

```
1 import static net.grinder.script.Grinder.grinder
2 import static org.junit.Assert.*
3 import static org.hamcrest.Matchers.*
4 import net.grinder.script.GTest
5 import net.grinder.script.Grinder
6 import net.grinder.scriptengine.groovy.junit.GrinderRunner
7 import net.grinder.scriptengine.groovy.junit.annotation.BeforeProcess
8 import net.grinder.scriptengine.groovy.junit.annotation.BeforeThread
9 // import static net.grinder.util.GrinderUtils.* // You can use this if you're using nGrinder after 3.2.3
10 import org.junit.Before
11 import org.junit.BeforeClass
12 import org.junit.Test
13 import org.junit.runner.RunWith
14
15 import org.ngrinder.http.HTTPRequest
16 import org.ngrinder.http.HTTPRequestControl
17 import org.ngrinder.http.HTTPResponse
18 import org.ngrinder.http.cookie.Cookie
19 import org.ngrinder.http.cookie.CookieManager
20
21 /**
22 * A simple example using the HTTP plugin that shows the retrieval of a single page via HTTP.
23 *
24 * This script is automatically generated by nGrinder.
25 */
26
27 You can use various log levels. [trace, debug, info, warn, error]
28 ex) grinder.logger.${level}("message")
29
30 You can access to response body with HTTPResponse.getText() method.
31 ex) HTTPResponse result = request.GET("...")
32     grinder.logger.debug(result.text)
33
34 You can test multiple transactions by recording new GTest instance.
35 ex) @BeforeProcess
36     public static void beforeProcess() {
37
38         static final Random rand = new Random()
39
40         String payload() {
41             Integer id    = rand.nextInt(60001 - 10001) + 10001
42             String name  = Long.toString(Math.abs(rand.nextLong()) % 3656158440062976L),
43
44             Integer height = rand.nextInt(190 - 165) + 165
45             Integer weight = rand.nextInt(100 - 46) + 46
46             Integer year  = rand.nextInt(2003 - 1970) + 1970
47             Integer mon   = rand.nextInt(12 - 1) + 1
48             Integer day   = rand.nextInt(31 - 1) + 1
49             Long now    = System.currentTimeMillis()
50
51             return "{ \"id\": \"" + id + "\", \"name\": \"" + name + "\", \"birthday\": \"" + year + "-" +
52                 mon + "-" + day + "\", \"height\": " + height + ", \"weight\": " + weight + ",\n \"timestamp\": " +
53                 "now\"}"
54         }
55
56         @Test
57         public void test() {
58             HTTPResponse response = request.POST("http://ngrinder-an1t-pub-alb-993527877.ap-northeast-1.elb.amazonaws.com/api/collect", payload().getBytes())
59
60             if (response.statusCode == 301 || response.statusCode == 302) {
61                 grinder.logger.warn("Warning. The response may not be correct. The response code was {}.", response.statusCode)
62             } else {
63                 assertEquals(response.statusCode, 201)
64             }
65         }
66     }
67 }
```

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

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

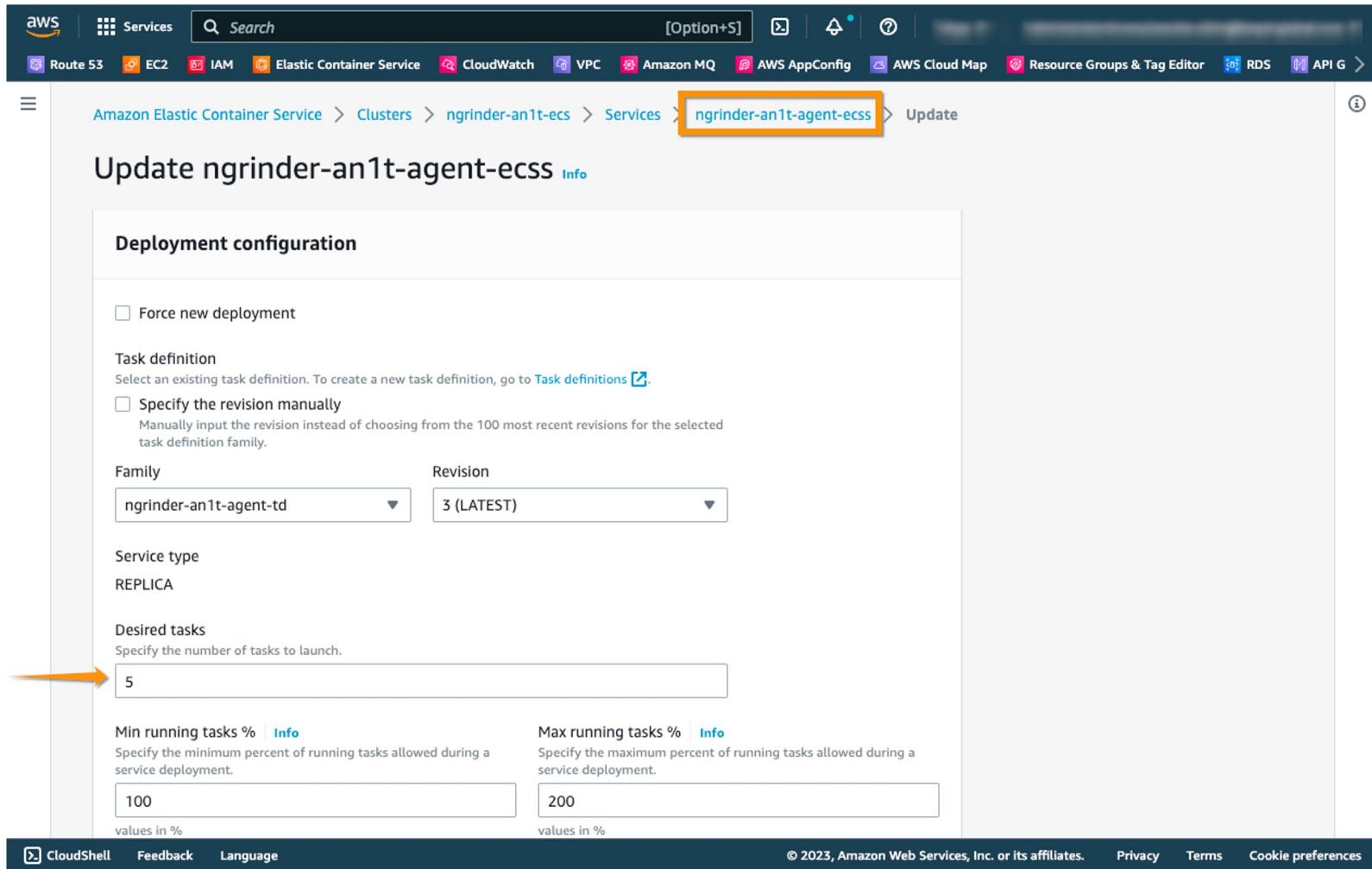
String payload() {
    Integer id    = rand.nextInt(60001 - 10001) + 10001
    String name  = Long.toString(Math.abs(rand.nextLong()) % 3656158440062976L),
36);
    Integer height = rand.nextInt(190 - 165) + 165
    Integer weight = rand.nextInt(100 - 46) + 46
    Integer year  = rand.nextInt(2003 - 1970) + 1970
    Integer mon   = rand.nextInt(12 - 1) + 1
    Integer day   = rand.nextInt(31 - 1) + 1
    Long now    = System.currentTimeMillis()
    return "{ \"id\": \"" + 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 {
        assertEquals(response.statusCode, 201)
    }
}
```

# nGrinder 애플리케이션 부하 규모 설정

앤타프라이즈용 대규모 부하를 발생하고 싶다면, agent 갯수를 원하는 만큼 조정하세요.



The screenshot shows the AWS ECS console with the path: Amazon Elastic Container Service > Clusters > ngrinder-an1t-ecs > Services > ngrinder-an1t-agent-ecss > Update. The 'Deployment configuration' section is displayed. A yellow arrow points to the 'Desired tasks' input field, which contains the value '5'. Other visible fields include 'Family' (ngrinder-an1t-agent-td), 'Revision' (3 (LATEST)), 'Service type' (REPLICA), 'Min running tasks %' (100), and 'Max running tasks %' (200).

Update ngrinder-an1t-agent-ecss [Info](#)

**Deployment configuration**

Force new deployment

**Task definition**  
Select an existing task definition. To create a new task definition, go to [Task definitions](#).

Specify the revision manually  
Manually input the revision instead of choosing from the 100 most recent revisions for the selected task definition family.

Family	Revision
ngrinder-an1t-agent-td	3 (LATEST)

**Service type**  
REPLICA

**Desired tasks**  
Specify the number of tasks to launch.

**Min running tasks %** [Info](#)  
Specify the minimum percent of running tasks allowed during a service deployment.  
 values in %

**Max running tasks %** [Info](#)  
Specify the maximum percent of running tasks allowed during a service deployment.  
 values in %

[CloudShell](#) [Feedback](#) [Language](#) © 2023, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

# nGrinder 테스트 시나리오 계획

가상의 사용자 수(부하량)을 조정하고 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 Performance Test Script

admin Help

List Test Name TEST01 Tags Please input tags. Save Save and Start

Description TEST01

Test Configuration

Basic Configuration

Agent 2 Max: 2 Vuser: 200 Enable Ramp-Up Thread

Vuser per agent 100 Max: 3000 Processes 2 Threads 50

Script svn TEST.groovy R HEAD

Script Resources

Target Host + Add

Duration 0 : 05 : 00 HH:MM:SS

Run Count 0 Max: 10000 Show Advanced Configuration

Connection reset on each test run Sampling Interval 2

Safe File Distribution Ignore Sample Count 0

Ignore Errors Parameter

Vuser Ramp-Up Chart per Agent

Time (s)	Vusers
0	0
1	2
2	4
3	6
4	8
5	10
6	12
7	14
8	16
9	18
10	20
11	22
12	24
13	26
14	28
15	30
16	32
17	34
18	36
19	38
20	40
21	42
22	44
23	46
24	48
25	50
26	52
27	54
28	56
29	58
30	60
31	62
32	64
33	66
34	68
35	70
36	72
37	74
38	76
39	78
40	80
41	82
42	84
43	86
44	88
45	90
46	92
47	94
48	96
49	98
50	100

# nGrinder 테스트 진행 및 모니터링

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

The image shows two side-by-side screenshots illustrating the monitoring of an nGrinder test execution.

**Left Side (nGrinder Interface):**

- Test Configuration:** Test Name: TEST01, Description: TEST01.
- Summary:** Total Users: 200 (Running 200), Total Processes: 4 (Running 4).
- Target Host:** Duration: 00:05:00 (HH:MM:SS), Runs 198,278.
- Agent State:** ip-172-76-22-213... CPU-24% MEM-31% RX-723.6K TX-1.58M  
ip-172-76-21-111... CPU-21% MEM-31% RX-687.4K TX-1.55M
- TPS Graph:** Shows throughput over time, starting around 1500 TPS and rising to approximately 3500 TPS by the end of the test.
- Accumulated Statistics:** Latest Sample: ID 1, Test Name: ngrinder-an1t-pub..., Success: 8,141, Errors: 0, MTT: 49, TPS: 4,071, MTTFB: 13, Resp/s: 0.0B.

**Right Side (AWS CloudWatch Metrics Dashboard):**

- Status:** ARN: ngrinder-an1t-ecs/ngrinder-an1t-collector-ecss, Status: Active.
- Tasks:** 0 Pending, 2 Running / 2 Desired.
- Health:** CPU utilization and Memory utilization graphs showing spikes during the test period.

# nGrinder 테스트 종료 및 리포팅

테스트가 종료되면 애플리케이션 성능 관련 주요 메트릭을 확인 할 수 있습니다. 테스트가 종료되면 애플리케이션이 scale-in 되겠죠 ^;, 덧붙여 S3에 적재된 데이터도 살펴 볼수 있습니다.

**nGrinder** Performance Test Script admin Help TIP

new Announcement  
To download agent, go to the right-top menu and select "download agents".

List Test Name TEST01 Tags Please input tags. Clone Clone and Start

Description TEST01

Test Configuration Report

Summary TPS Graph Detailed Report

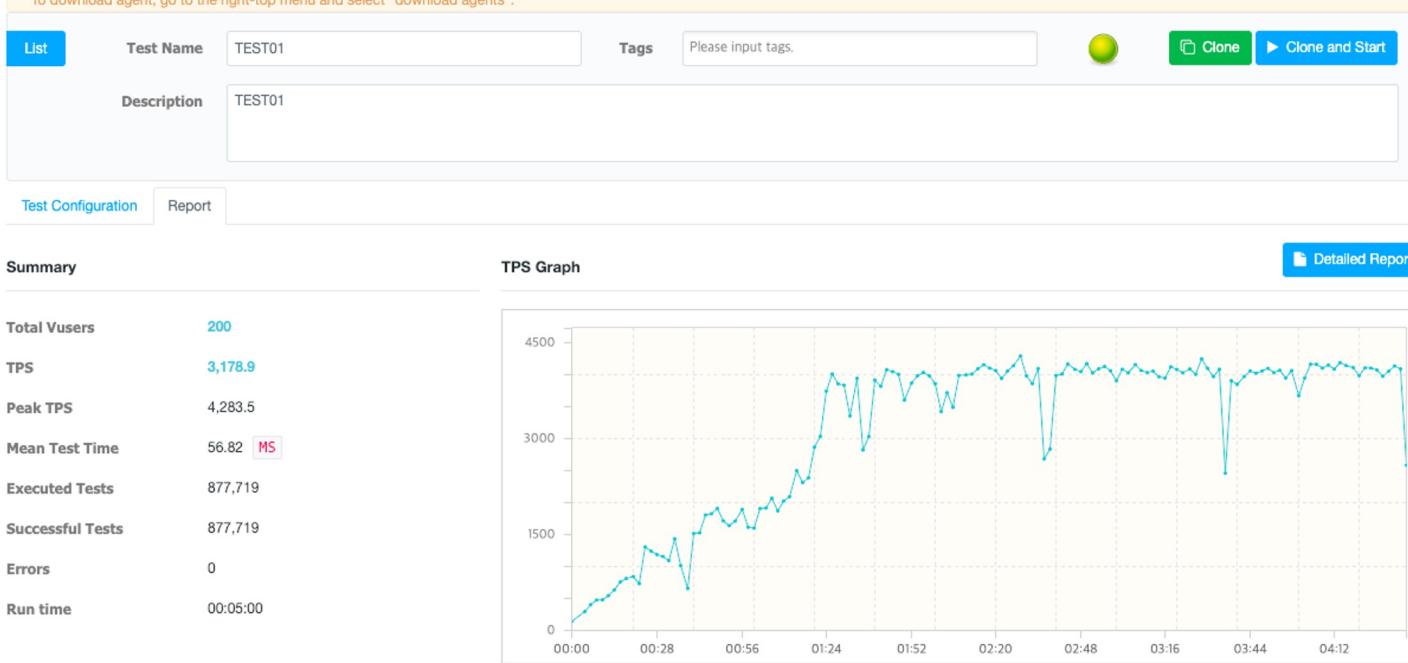
Total Vusers	200
TPS	3,178.9
Peak TPS	4,283.5
Mean Test Time	56.82 MS
Executed Tests	877,719
Successful Tests	877,719
Errors	0
Run time	00:05:00

Logs Leave Comment

ip-172-76-21-111.ap-northeast-1.compute.internal--log.zip  
ip-172-76-22-213.ap-northeast-1.compute.internal--log.zip

Comment Leave Comment

nGrinder v3.5.5

The screenshot shows the nGrinder test summary page. At the top, there's a navigation bar with 'nGrinder' logo, 'Performance Test', 'Script', 'admin', and 'Help'. A 'TIP' button is also present. Below the navigation, there's a message about downloading agents. The main area has tabs for 'List', 'Test Name' (set to 'TEST01'), 'Tags' (input field), 'Clone' (button), and 'Clone and Start' (button). There's also a 'Description' field containing 'TEST01'. Below these are 'Test Configuration' and 'Report' buttons. The 'Summary' section on the left lists various metrics: Total Vusers (200), TPS (3,178.9), Peak TPS (4,283.5), Mean Test Time (56.82 MS), Executed Tests (877,719), Successful Tests (877,719), Errors (0), and Run time (00:05:00). To the right is a 'TPS Graph' showing a fluctuating line over time from 00:00 to 04:12. A 'Detailed Report' button is located above the graph. At the bottom, there are sections for 'Logs' (with two log files listed) and 'Comment' (with a 'Leave Comment' button). The footer indicates the version 'nGrinder v3.5.5'.

# Conclusion

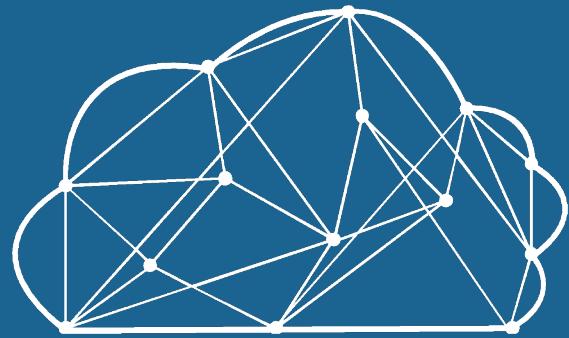
---

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 규모에서부터 대규모 엔터프라이즈 환경까지 대응 하는 부하를 원하는 시간에 원하는 만큼 발생 할 수 있으며, 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