



CloudWatch

1. 개념

CloudWatch

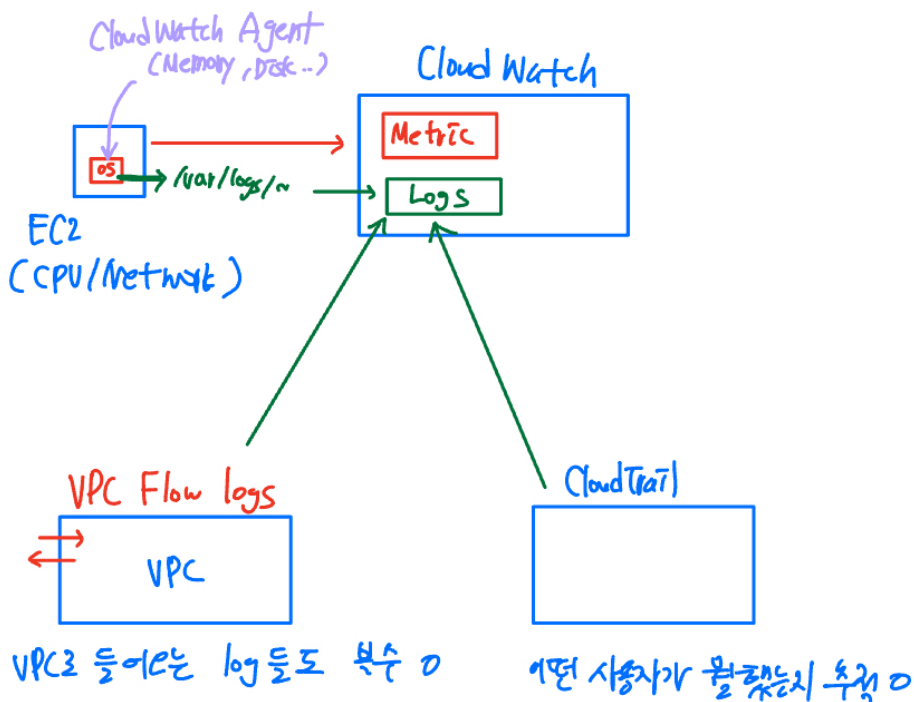
- 거의 실시간에 가까운 시스템 이벤트 스트림을 제공하는 서비스
- CloudWatch 사용 시 운영 변경 사항에 신속하게 대응 가능

CloudWatch 경보

- 알람을 보내거나 or 정의한 규칙을 기준으로 모니터링하는 리소스를 자동으로 변경함

CloudWatch Agent

- AWS의 모니터링 서비스인 Amazon CloudWatch와 함께 사용되는 소프트웨어
- Cloudwatch에서 기본으로 제공하는 모니터링 외에 추가로 **메모리, 디스크 사용량 등 더 많은 지표들을 수집**하려면 Cloudwatch Agent를 설치해야 됨



원래 CloudWatch는 내부의 메모리 등의 정보는 보지 X

(∵ 객체의 OS 안의 정보는 함부로 볼 수 X)

but 객체 CloudWatch Agent를 설치하면 OS 안의 메모리 정보 (Disk) 또한 볼 수 O.
역할부여 (IAM) 통해 볼 수 O.

OS 안에 있는 log 또한 CloudWatch 통해 확인 O



CloudWatch vs CloudTrail 차이점

CloudWatch

- 퍼포먼스 체크
- AWS의 서비스 뿐만 아니라 어플리케이션의 로그 및 동작 로그 취합
- 어플리케이션이 어떻게 동작했는지, 무슨 버그가 있는지, 메모리는 얼마나 소모되었는지
- 대시보드 및 알람 등 모니터링을 위한 서비스 제공

CloudTrail

- 감시
- AWS의 모든 서비스가 사용될 때마다 사용 로그 저장
- AWS가 언제 어디서 누구에 의해 사용되었는지
- 단순히 AWS 사용 로그만 저장

2. 기능

1. 로그

- AWS 내외의 로그를 모아 보관 & 이를 사용자에게 전달
- 주요 서비스에 대한 모니터링(로그, 메트릭 등) 제공
 - EC2, ASG, ELB, Route 53
 - CloudFront, EBS, Storage GW 등
- 로그를 쿼리 형식으로 분석가능한 Insight 활용 가능

2. 경보

- 로그를 기반으로 지표를 생성해서 특정 지표의 조건에 따라 경보 발생
 - 경보는 다른 서비스 호출 가능
- ex) cpu 사용량이 일정 수준이라면, 호출 람다에 에러가 발생한다면

3. 이벤트

- 일정 주기 or AWS의 여러 이벤트를 감지해 다른 AWS 서비스(SNS, Lambda 등)를 호출하는 규칙
 - Event Bridge 규칙과 동일
- 일정 주기로 이벤트 생성 가능
 - ex) 매시 정각마다 하루에 쌓인 로그 분석



Skill Builder : EC2 Instance Rightsizing (Korean)- 실습완료

실습 개요

Amazon CloudWatch 지표를 기반으로 더 적합한 구성을 채택할 수 있도록 두 **EC2 인스턴스의 비용 최적화 규모 조정하기**

그런 다음, 각 서버의 인스턴스 유형이나 인스턴스 크기를 수정하는 동시에 **향후 리소스 제약 조건을 모니터링하도록 CloudWatch 경보 구성하기**

목표

- Amazon EC2 인스턴스에 **CloudWatch 에이전트**를 설치하고 구성합니다.
- Amazon EC2 인스턴스에서 대화형 명령을 실행하여 **CPU 및 메모리 로드**를 시뮬레이션합니다.
- CloudWatch 대시보드를 사용하여 활용된 Amazon **EC2 인스턴스 지표**를 검토합니다.
- 관찰된 CloudWatch 지표를 기반으로 **Amazon EC2 인스턴스 크기 또는 유형**을 수정합니다.
- **CloudWatch 경보**를 구성합니다.

실습 환경

다음은 실습 시작 시 두 서버 환경의 비용을 보여주는 AWS 요금 계산기 추정치 스크린샷임

AWS Pricing Calculator > Before Right Sizing

Before Right Sizing Export

Estimate date: September 7, 2023

Estimate summary Info			Getting Started with AWS
Upfront cost 0.00 USD	Monthly cost 244.91 USD	Total 12 months cost 2,938.92 USD Includes upfront cost	Get started for free Contact Sales

Before Right Sizing

<input type="checkbox"/>	Service Name	Upfront cost	Monthly cost	Description	Region	Config Summary
<input type="checkbox"/>	Amazon EC2	0.00 USD	122.20 USD	Before Right Sizing	US West (Oregon)	Tenancy (Shared Instances), Operating system (Linux), Workload (Consistent, Number of instances: 1...
<input type="checkbox"/>	Amazon EC2	0.00 USD	122.11 USD	Before Right Sizing	US West (Oregon)	Tenancy (Shared Instances), Operating system (Linux), Workload (Consistent, Number of instances: 1...
<input type="checkbox"/>	Amazon CloudWatch	0.00 USD	0.60 USD	Before Right Sizing	US West (Oregon)	Number of Metrics (includes detailed and custom metrics) (2), Number of Dashboards (1), Number of...

태스크 1: Web-Application-Instances 리소스 그룹에 CloudWatch 에이전트 설치 및 구성

팀과의 논의를 통해 메모리 사용률이 웹 애플리케이션 내 성능 문제를 진단하는 핵심 지표라는 것을 알게 되었음

하지만 메모리 사용률은 CloudWatch가 EC2 인스턴스에 직접 설치된 CloudWatch 에이전트(CWA) 없이 수집할 수 없음

CWA ⇒ 온프레미스 및 대부분의 클라우드에서 서버에 대한 통계 및 지표를 수집하는 데 사용할 수 있으므로 하이브리드 환경에 유용한 도구임

이 태스크에서는 웹 애플리케이션 인스턴스에 CloudWatch 에이전트를 설치함

AWS Systems Manager를 사용하면 이 실습을 위해 이미 생성된 리소스 그룹을 사용하여 두 EC2 인스턴스에 대한 설치 및 구성 명령을 동시에 실행할 수 있음

태스크 1.1: 웹 애플리케이션 AMAZON EC2 인스턴스에 CLOUDWATCH 에이전트 설치

- Systems Manager > Run Command > Run Command 버튼 선택
- 다음과 같이 구성하기
 - **Command document** 섹션
 - AWS-ConfigureAWSPackage 검색하고 선택
 - **Command parameters** 섹션
 - Action에서 Install을 선택

- Installation Type에서 Uninstall and reinstall을 선택



Uninstall and reinstall & In-place update

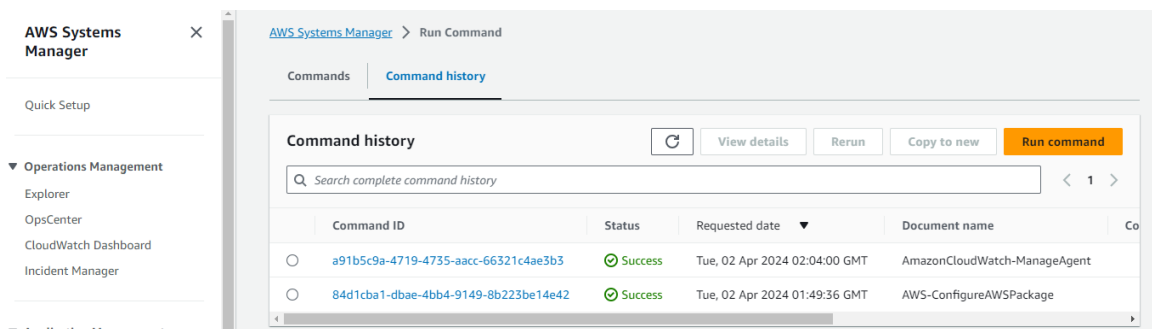
- **Uninstall and reinstall**(제거 및 재설치): 재설치 프로세스가 완료될 때까지 응용 프로그램을 오프라인으로 가져감
- **In-place update**(위치 변경 업데이트): 새로운 또는 업데이트된 파일이 설치에 추가될 때까지 응용 프로그램을 사용할 수 있음

- Name에 AmazonCloudWatchAgent를 입력
- **Target selection** 섹션
 - Choose a resource group 선택
 - Web-Application-Instances 리소스 그룹은 현재 유일한 리소스 그룹이므로 자동으로 선택됨
 - 그렇지 않은 경우 Resource group 드롭다운 메뉴에서 Web-Application-Instances 리소스 그룹을 수동으로 선택하기
- **Output options** 섹션
 - Enable an S3 bucket 옵션 선택 취소

태스크 1.2: WEB-APPLICATION-INSTANCES 리소스 그룹에서 CLOUDWATCH AGENTS 구성 및 시작

- 왼쪽 탐색 창의 Run Command > Run Command > Run Command 버튼 선택
- 다음과 같이 구성하기
 - **Command document** 섹션
 - AmazonCloudWatch-ManageAgent 검색 후 선택
 - **Command parameters** 섹션
 - Action 에서 configure 선택
 - Optional Configuration Source에서 ssm을 선택
 - Optional Configuration Location에 AmazonCloudWatch-AgentConfig를 입력

- Optional Restart에서 yes를 선택
- **Target selection** 섹션
 - Choose a resource group 선택
- **Output options** 섹션
 - Enable an S3 bucket 옵션을 선택 취소
- 지금까지 Web-Application-Instances 리소스 그룹에 CloudWatch 에이전트를 성공적으로 설치하고 구성한 것임

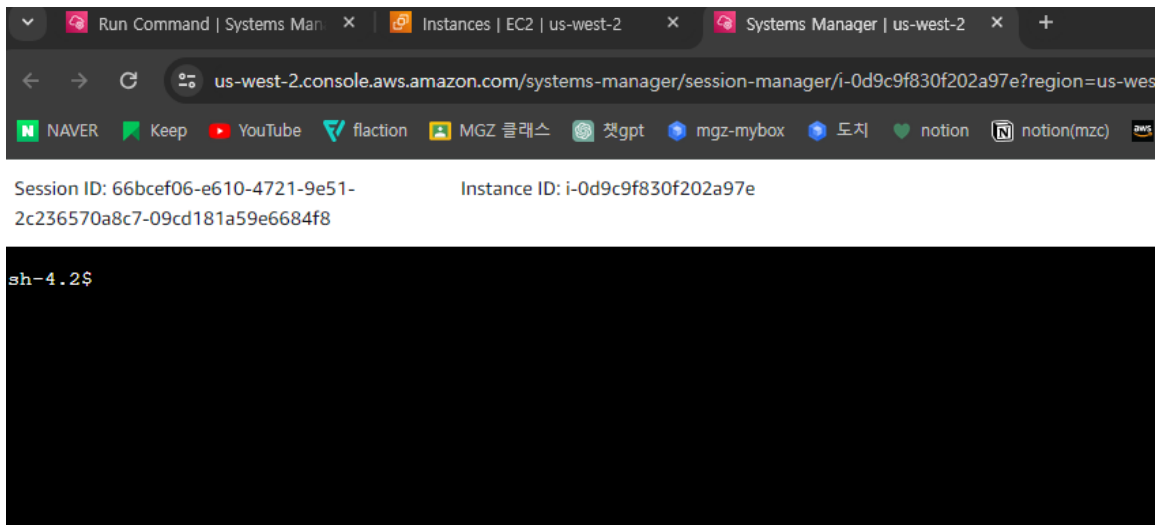


태스크 2: 웹 서버 스트레스 테스트 및 CPU 지표 분석

EC2 인스턴스에 CloudWatch 에이전트를 설치하고 구성했으므로 **활용도를 시뮬레이션**하고 **성능 지표를 조사**할 차례임

웹 서버 EC2 인스턴스를 로드하고 스트레스를 주는 도구인 **stress-ng**를 사용함

- *stress-ng* 도구는 초기 랩 프로비저닝 중에 웹 서버 인스턴스에 이미 설치되어 있음
- EC2 > Instances > Web-Server 인스턴스를 선택 > Connect > Session Manager 탭 > Connect



- *stress-ng* 도구 설치 확인하기

```
cd ~ && stress-ng --version
```

```
sh-4.2$ cd ~ && stress-ng --version
stress-ng, version 0.07.29
sh-4.2$
```

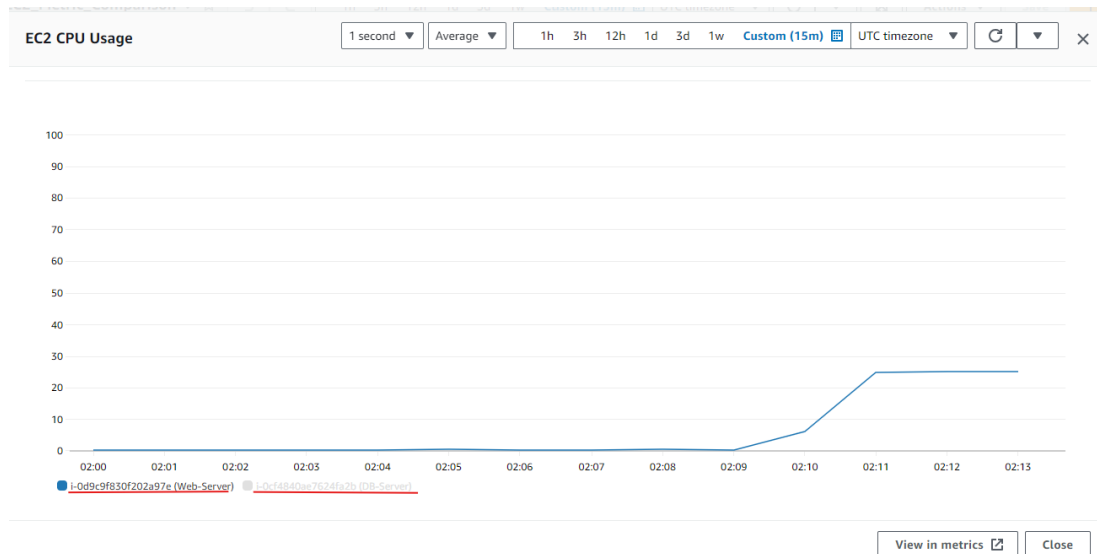
- *stress-ng*를 사용하여 웹 서버 인스턴스의 애플리케이션 워크로드를 시뮬레이션하여 높은 로드 조건에서 응답하는 방식을 테스트하기
 - Web-Server 인스턴스 로드하고 스트레스를 가하기

```
stress-ng --cpu 1stress-ng --cpu 1
```

```
sh-4.2$ stress-ng --cpu 1
stress-ng: info:  [6427] defaulting to a 86400 second run per stressor
stress-ng: info:  [6427] dispatching hogs: 1 cpu
```

- 이 명령은 이 실습에서 웹 서버의 활용도를 시뮬레이션하는 하나의 작업자 스레드로 CPU를 테스트함
 - 이 브라우저 탭을 열어두기
 - 탭 닫을 시 실행 중인 *stress-ng* 프로세스와 함께 세션이 종료됨
- Web-Server 인스턴스의 CPU 사용률을 검토하고 CloudWatch를 사용하여 분석하기

- CloudWatch > Dashboards를 선택 > EC2_Metric_Comparison 대시보드 링크를 선택
- EC2 CPU Usage 그래프 위로 마우스를 가져가서 Maximize 버튼을 선택하여 그래프를 확장하기
- 선택적으로 인스턴스 하나를 선택해 해당 그래프에서 필터링 가능함



- *stress-ng*를 사용하여 로드를 시뮬레이션한 후 웹 서버 인스턴스의 CPU 사용량은 25%에 가까움
 - 이는 --cpu 1로 단일 작업자 스레드를 지정했고, t3.xlarge 인스턴스 크기가 4개의 vCPU를 제공하기 때문임
- 현재 실행 중인 프로세스를 중지하기
 - 세션 관리자 터미널 창으로 돌아가서 CTRL+C

⇒ 25%의 CPU 사용률을 보고하는 워크로드를 통해 **Web-Server** 인스턴스가 **CPU 영역**에서 **과도하게 프로비저닝**되었다는 결론을 내릴 수 있음

⇒ 더 적은 CPU 리소스를 제공하는 대신 더 작은 인스턴스 크기를 선택하면 t3.xlarge의 지속적인 비용을 최적화할 수 있음

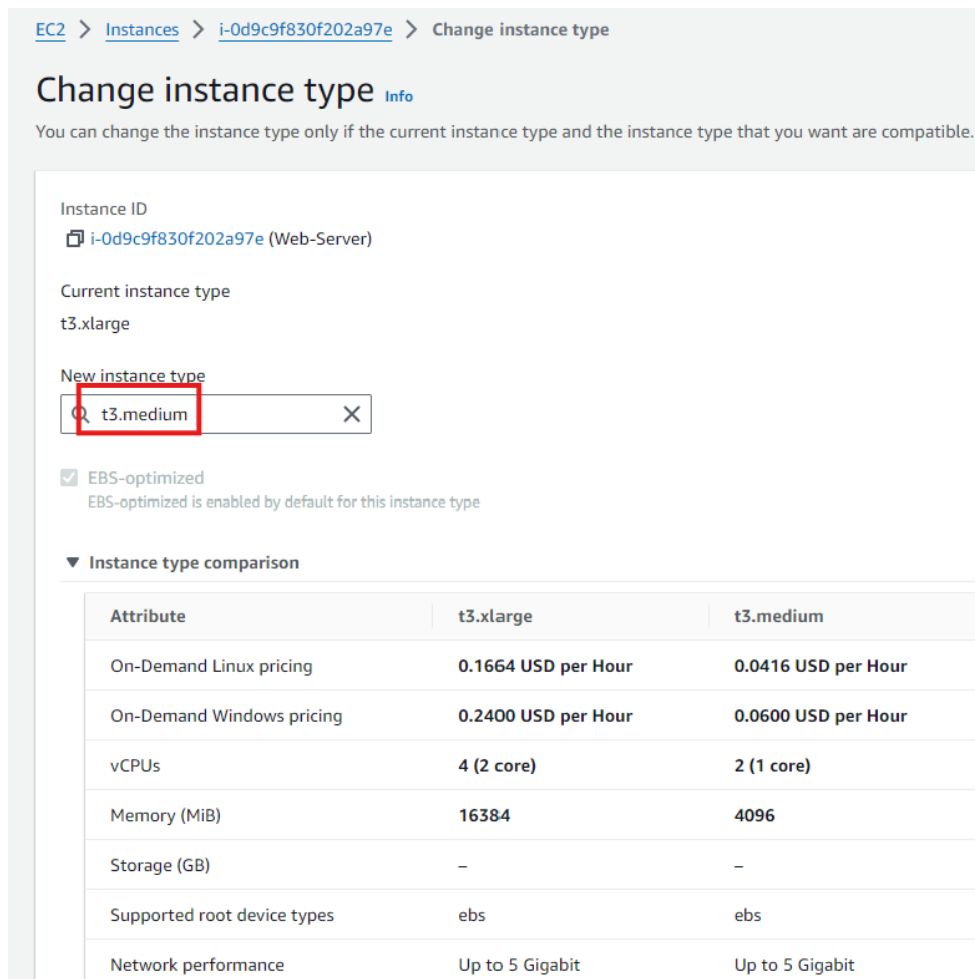
태스크 3: CPU 지표를 분석하기 위해 웹 서버의 크기를 조정하고 스트레스 테스트 수행

웹 서버의 크기를 조정하고 CPU 지표를 다시 분석하여 크기 조정 작업을 검증하기

태스크 3.1 WEB-SERVER 크기 조정

CPU 지표를 분석한 후 CPU 사용량이 평균적으로 낮고 최고(최대)에 있다는 것을 보고하므로 Web-Server 인스턴스의 크기를 4개의 vCPU 대신 **2개의 vCPU가 있는 t3.medium**으로 조정하기로 결정함

- CloudWatch 대시보드의 EC2 CPU Usage가 표시된 브라우저 탭을 유지하고, AWS Management Console이 있는 브라우저 탭으로 다시 전환함
- EC2 > Instances > Web-Server 인스턴스 선택 > Instance state 선택 > Stop instance 선택
- Web-Server 인스턴스 선택 > Instance settings 선택 > Change instance type 선택 > t3.medium으로 변경 > Apply



EC2 > Instances > i-0d9c9f830f202a97e > Change instance type

Change instance type [Info](#)

You can change the instance type only if the current instance type and the instance type that you want are compatible.

Instance ID
i-0d9c9f830f202a97e (Web-Server)

Current instance type
t3.xlarge

New instance type
t3.medium

☒ EBS-optimized
EBS-optimized is enabled by default for this instance type

▼ Instance type comparison

Attribute	t3.xlarge	t3.medium
On-Demand Linux pricing	0.1664 USD per Hour	0.0416 USD per Hour
On-Demand Windows pricing	0.2400 USD per Hour	0.0600 USD per Hour
vCPUs	4 (2 core)	2 (1 core)
Memory (MiB)	16384	4096
Storage (GB)	–	–
Supported root device types	ebs	ebs
Network performance	Up to 5 Gigabit	Up to 5 Gigabit

- Web-Server 인스턴스 선택 > Instance state 선택 > Start instance 선택

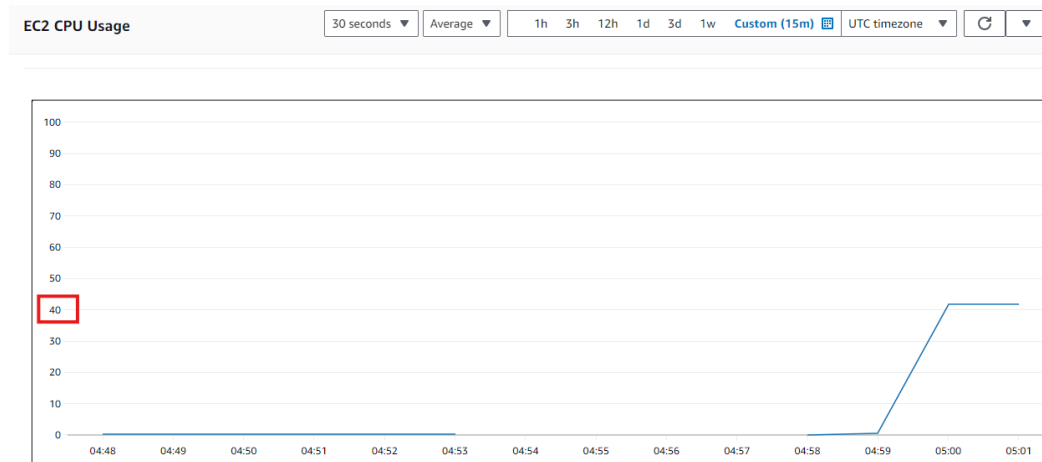
태스크 3.2 WEB-SERVER 스트레스 테스트를 다시 수행하여 CPU 지표 분석

- Web-Server 인스턴스 선택 > Session Manager 탭 선택 > Connect 선택 > 다시 stress-ng 사용하기

```
cd ~ && stress-ng --cpu 1
```

```
sh-4.2$ cd ~ && stress-ng --cpu 1
stress-ng: info:  [2414] defaulting to a 86400 second run per stressor
stress-ng: info:  [2414] dispatching hogs: 1 cpu
```

- Web-Server에 대한 CloudWatch 대시보드 EC2 CPU Usage를 다시 검토하고 CloudWatch를 사용하여 분석하기
 - CloudWatch > Dashboards를 선택 > EC2_Metric_Comparison 대시보드 링크를 선택
 - EC2 CPU Usage 확인
 - cpu 사용량이 40%이상임



태스크 4: CPU 사용률 경고 생성

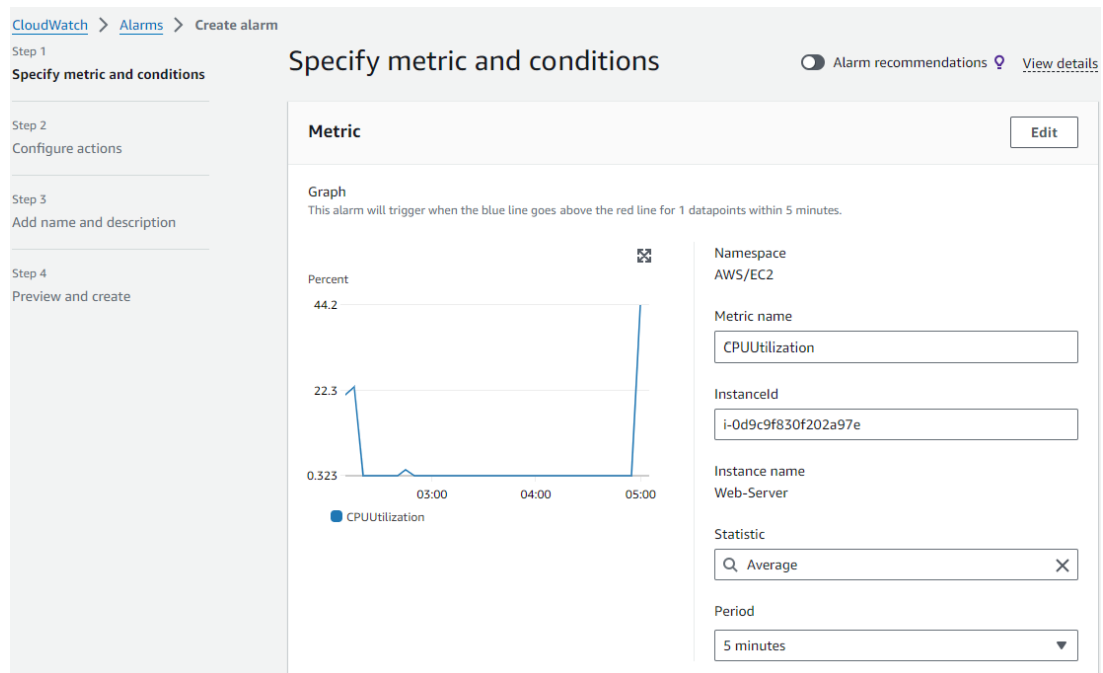
웹 서버의 인스턴스 크기를 변경했으므로 이제 **CPU 사용률 지표를 모니터링**하여 나중에 인스턴스의 사용률이 지나치게 높아지거나 낮아지지 않도록 하는 것이 중요함

이를 용이하게 하기 위해 CPU 사용률을 모니터링하도록 **CloudWatch 경고 설정**하기

태스크 4.1: CPU 사용률 과다 경고

- CloudWatch > 왼쪽 탐색 창의 Alarms > All Alarms > Create alarm

- **specify metric and conditions** 페이지
 - Select metric 선택
- **AWS namespaces** 섹션
 - EC2 카드 선택 > Per-Instance Metrics 카드를 선택 > 검색창에 CPUUtilization 입력 후 Enter > 인스턴스 목록에서 Web-Server 인스턴스를 선택 > Select metric 선택



- **Conditions** 섹션

Conditions

Threshold type

☒ Static
Use a value as a threshold

☐ Anomaly detection
Use a band as a threshold

Whenever CPUUtilization is...
Define the alarm condition.

☐ Greater
> threshold

☒ Greater/Equal
≥ threshold

☐ Lower/Equal
≤ threshold

☐ Lower
< threshold

than...
Define the threshold value.

75

Must be a number

► Additional configuration

- Threshold type에 Static을 선택
- Whenever CPUUtilization is... 에서 Greater/Equal을 선택
- than... 에 75를 입력
- Configure actions 페이지의 Notification 섹션에서 Remove를 선택
- Add name and description 페이지에서 Alarm name에 Over-utilized CPU: Web-Server 입력하기 > Create alarm

☑ Successfully created alarm **Over-utilized CPU: Web-Server.**

CloudWatch > Alarms

Alarms (1) Hide Auto Scaling alarms Clear selection ↻

Q Search Alarm state: Any ▼ Alarm type: Any

<input type="checkbox"/>	Name ▼	State ▼	Last state update ▼	Conditions	Actions
<input type="checkbox"/>	Over-utilized CPU: Web-Server	⚠ Insufficient data	2024-04-02 05:12:55	CPUUtilization ≥ 75 for 1 datapoints within 5 minutes	No actions

태스크 4.2: CPU 사용률 부족 경보

CPU 사용률 부족이 애플리케이션 성능과 관련하여 반드시 우려할 문제는 아니지만 **과도하게 프로비저닝된 EC2 인스턴스를 빠르게 식별하기 위해 경고를 생성하는 것이 좋음**

- Create alarm
- **Specify metric and conditions** 페이지
 - Select metric 선택

- **AWS namespaces** 섹션
 - EC2 카드를 선택 > Per-Instance Metrics 카드를 선택 > 검색 표시줄에 CPUUtilization을 입력하고 Enter 키 > Web-Server 인스턴스를 선택 > Select metric을 선택
- **Conditions** 섹션
 - Threshold type에 Static을 선택
 - Whenever CPUUtilization is... 에 Lower/Equal을 선택
 - than... 에 25를 입력
- **Configure actions** 페이지의 Notification 섹션에서 Remove를 선택
- **Add name and description** 페이지
 - Alarm name에 Under-utilized CPU: Web-Server를 입력
- 결과

☑ Successfully created alarm Under-utilized CPU: Web-Server.

CloudWatch > Alarms

Alarms (2) ☐ Hide Auto Scaling alarms

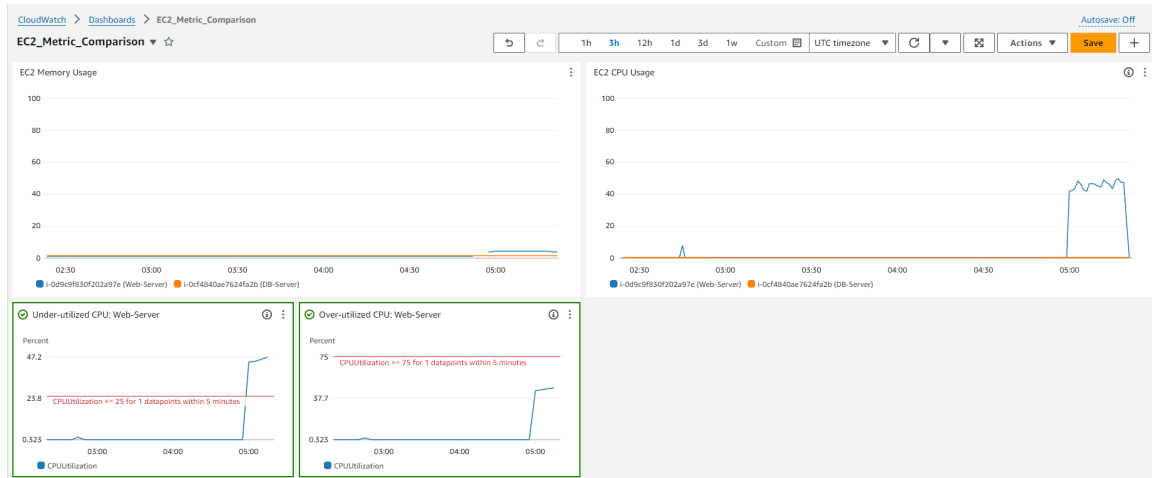
Q Search Alarm state: Any Alarm type: Any Actions status: Any

<input type="checkbox"/>	Name	State	Last state update	Conditions	Actions
<input type="checkbox"/>	Under-utilized CPU: Web-Server	OK	2024-04-02 05:17:35	CPUUtilization <= 25 for 1 datapoints within 5 minutes	No actions
<input type="checkbox"/>	Over-utilized CPU: Web-Server	OK	2024-04-02 05:13:57	CPUUtilization >= 75 for 1 datapoints within 5 minutes	No actions

태스크 4.3: 대시보드에 경보 추가

두 개의 경보를 생성했으므로 이제 애플리케이션 상태에 대한 단일 뷰의 개발을 시작하기 위해 경보를 대시보드에 추가하기

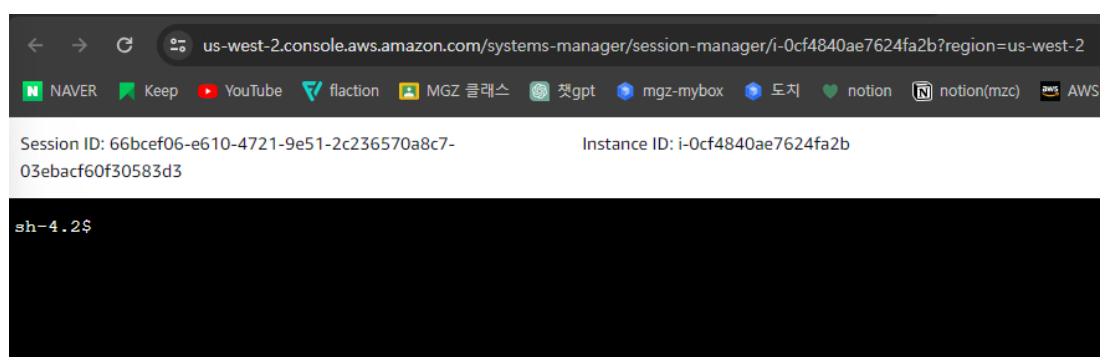
- 방금 생성한 두 경보를 선택 > Actions > Add to dashboard > 팝업 창에서 Select a dashboard 검색 표시줄을 사용하고 EC2_Metric_Comparison을 선택 > Add to dashboard > 화면 상단의 View dashboard를 선택하여 대시보드로 이동 > save 입력



태스크 5: DB 서버 로드 테스트 및 메모리 지표 분석

head 명령을 사용하여 메모리 사용률을 시뮬레이션한 다음, DB-Server 인스턴스에 대한 메모리 지표 분석하기

- EC2_Metric_Comparison CloudWatch 대시보드를 열어 두고 AWS Management Console이 있는 브라우저 탭으로 다시 전환하기
- EC2 > Instances > DB-Server 인스턴스를 선택 > Connect > Session Manager 탭 > Connect
 - 새 브라우저 탭 또는 창이 열리고 DB-Server 인스턴스에 대한 연결이 표시됨

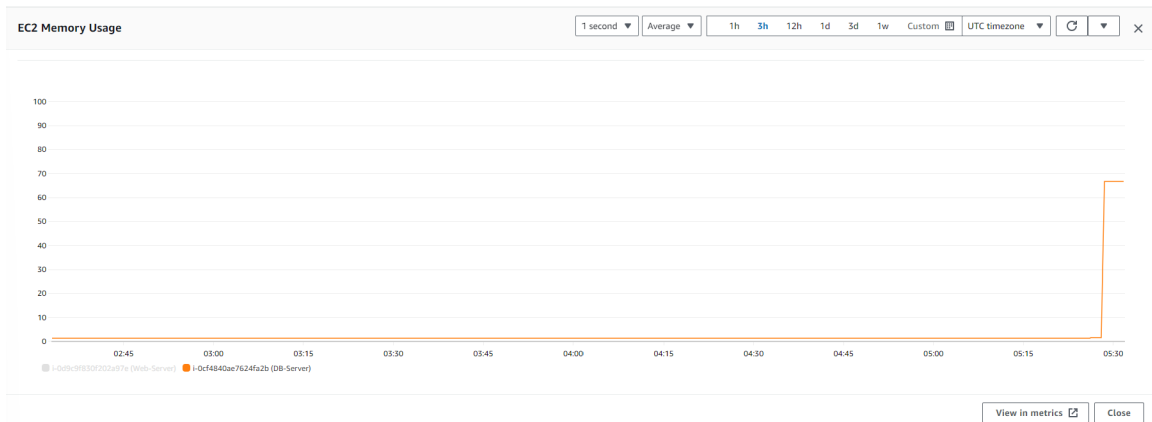


- DB-Server 인스턴스에서 메모리 사용률 시뮬레이션하기

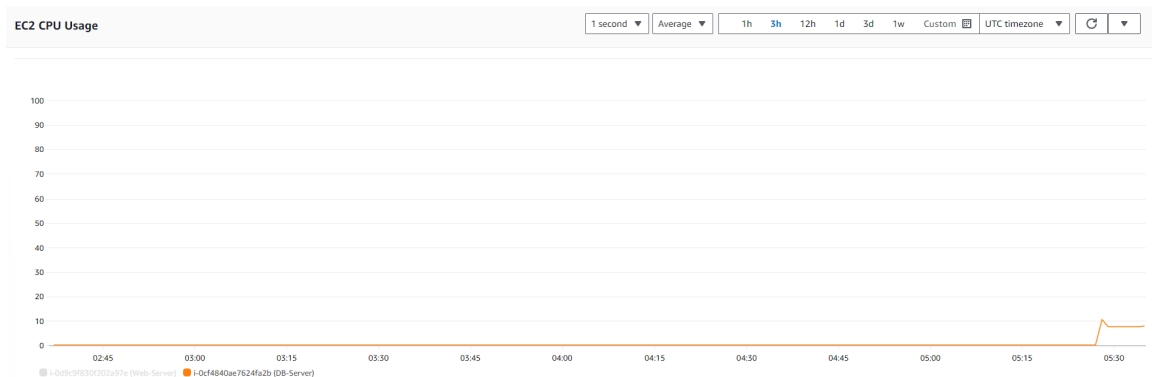
```
head -c 10G /dev/zero | tail
```

- /dev/zero에서 10GB의 정보를 할당하고, 이를 tail 프로그램으로 전송하여 이 명령에서 메모리 활용을 시뮬레이션하는 것

- CloudWatch를 사용하여 DB-Server 인스턴스의 메모리 사용률 검토해보기
- EC2_Metric_Comparison CloudWatch 대시보드로 다시 전환하기
- EC2 Memory Usage 위젯 내에서 DB-Server EC2 인스턴스 항목을 강조 표시하여 해당 메모리 지표를 검토하기



- head 및 tail 명령으로 메모리 사용량을 시뮬레이션한 결과, DB 서버 인스턴스의 메모리 사용량은 70%에 가까움
- EC2 CPU Usage 위젯 내에서 DB-Server EC2 인스턴스 항목을 강조 표시하여 해당 메모리 지표를 검토하기



⇒ DB-Server 인스턴스가 메모리에 대해서는 적절하게 프로비저닝되었지만, CPU에 대해서는 과도하게 프로비저닝되었다는 결론을 내릴 수 있음

태스크 6: DB-Server 인스턴스 최적화 및 메모리 지표 분석

DB-Server 지표를 분석한 후 메모리 사용량이 더 작은 인스턴스 크기에 맞지 않지만 CPU 사용량은 최소화되어 크기를 조정할 수 있다고 보고했음

이는 DB 서버의 용도에 맞는 더 나은 인스턴스 유형을 찾는 잠재적인 근거임

태스크 6.1 적합한 인스턴스 유형 검토

DB-Server 인스턴스에 더 적합한 인스턴스 유형을 찾는 방법은 여러 가지가 있음

여기서는 EC2 콘솔 내의 Instance Types 페이지를 활용하여 DB 인스턴스에 대해 관찰한 사용 패턴에 따라 리소스 범위를 찾음

- AWS Management Console이 있는 브라우저 탭으로 다시 전환하기
- EC2 > 왼쪽 탐색 창에서 Instance Types를 선택
 - DB-Server 인스턴스에 가장 효율적인 제품을 찾으려고 함
- 검색창에 다음 필터 입력
 - Architecture = x86_64
 - vCPUs >= 2
 - Memory (GiB) = 16
- 오른쪽 톱니바퀴를 눌러 Instance types 테이블의 Preferences 메뉴 열기 > On Demand Linux Pricing 옵션을 찾아서 활성화 > confirm

Preferences

Page size

☐ 10 Instance types
 ☐ 25 Instance types
 ☒ 50 Instance types

☐ Wrap lines

If checked, text in table will continue onto the next line when there isn't enough space. If unchecked, text will be truncated if there isn't enough space.

☐ Use regular expression matching

Enables regular expression support for client filters in the table instead of partial string match

☒ Context menu

Replace the browser context menu on table rows to allow quick actions.

☒ Row click selection

Enable to allow clicking anywhere on a row item to select the item

Attribute columns

Choose instance type attributes to display as columns in the table

☐ On-Demand us-west-2-pdx-1 Linux pricing
 ☐ On-Demand us-west-2-pdx-1 RHEL pricing
 ☐ On-Demand us-west-2-pdx-1 SUSE pricing
 ☐ On-Demand us-west-2-pdx-1 Windows pricing
 ☐ On-Demand us-west-2-sea-1 SUSE pricing
 ☐ On-Demand us-west-2-sea-1 RHEL pricing
 ☒ On-Demand us-west-2-sea-1 Linux pricing
 ☐ On-Demand us-west-2-sea-1 Windows pricing

Cancel

Confirm

- **On-Demand Linux Pricing** 열을 선택하여 **On-Demand Linux pricing**을 기준으로 정렬하기

Instance types (50)

Instance type

Insta...

Insta...

Hyperv...

vCPUs

Architecture

Memory (GiB)

Storage (GiB)

Storage type

On-Demand Linux pricing

<input type="checkbox"/> r5a.large	r5a	large	nitro	2	x86_64	16	-	-	0.113 USD per Hour
<input type="checkbox"/> r6a.large	r6a	large	nitro	2	x86_64	16	-	-	0.1134 USD per Hour
<input type="checkbox"/> r5.large	r5	large	nitro	2	x86_64	16	-	-	0.126 USD per Hour
<input type="checkbox"/> r6i.large	r6i	large	nitro	2	x86_64	16	-	-	0.126 USD per Hour
<input type="checkbox"/> r5ad.large	r5ad	large	nitro	2	x86_64	16	75	ssd	0.131 USD per Hour
<input type="checkbox"/> r5d.large	r5d	large	nitro	2	x86_64	16	75	ssd	0.144 USD per Hour

- 가장 효율적으로 가격이 책정된 인스턴스 옵션이 위에서 아래로 표시됨

태스크 6.2: DB-SERVER 수정

사용 가능한 모든 옵션을 검토한 후 DB-Server 인스턴스에 대해 *r5.large* 메모리 최적화 인스턴스 유형을 선택하기로 결정함

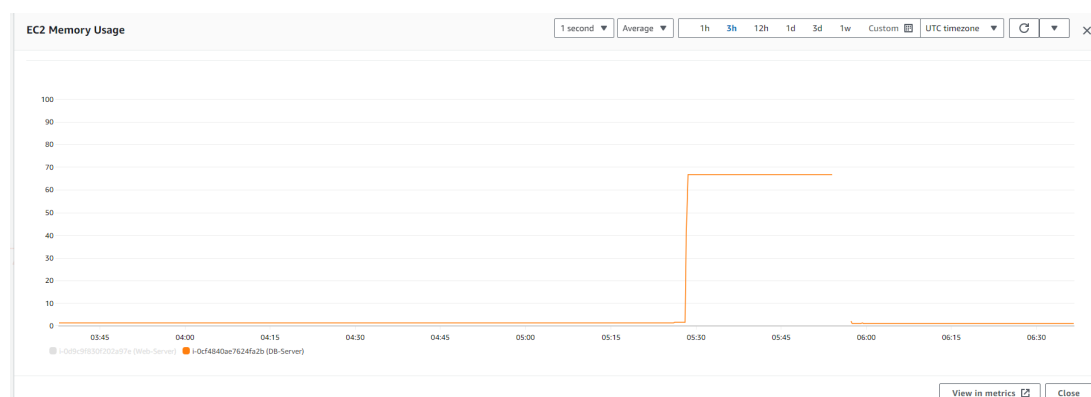
- **Instances** 섹션에서 **Instances**를 선택 > **DB-Server** 인스턴스를 선택 > **Instance state** 를 선택 > **Stop instance**
- **DB-Server** 인스턴스 중지하기
- **r5.large**로 변경하기
- **DB-Server** 인스턴스 다시 시작하기

태스크 6.3: DB-SERVER 인스턴스의 메모리 활용도를 시뮬레이션하고 메모리 지표를 다시 분석

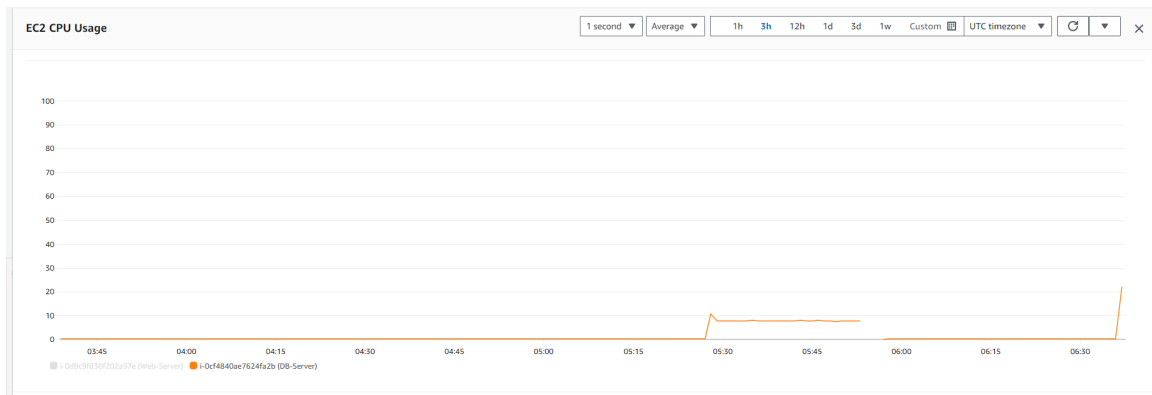
- DB-Server 인스턴스를 Session Manager로 연결하기
- DB-Server 인스턴스에서 메모리 사용률을 시뮬레이션하기

```
head -c 10G /dev/zero | tail
```

- B-Server 인스턴스의 메모리 사용률을 다시 검토하고 CloudWatch를 사용하여 분석해 보기
 - **EC2_Metric_Comparison** CloudWatch 대시보드 > **EC2 Memory Usage** 위젯 내에서 **DB-Server** EC2 인스턴스 항목을 강조 표시하여 해당 메모리 지표를 검토



- DB-Server 인스턴스의 EC2 CPU Usage 확인하기



⇒ 메모리 및 CPU 지표를 관찰한 후 이제 크기 조정 작업 후에 메모리와 CPU 모두에 대해 **DB-Server** 인스턴스가 적절하게 프로비저닝되었다는 결론을 내릴 수 있음

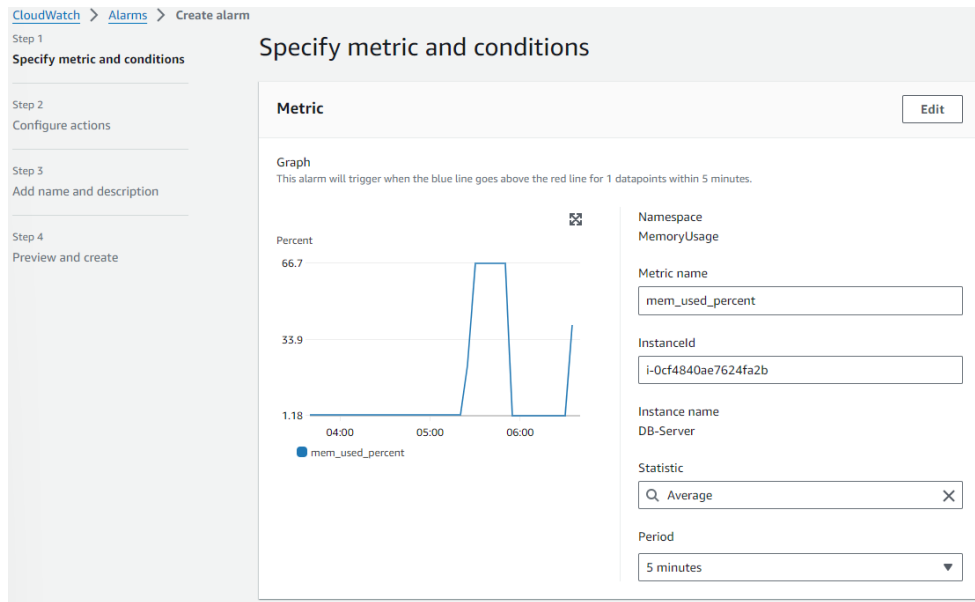
태스크 7: 메모리 사용률 경고 생성

이제 DB 서버 인스턴스를 최적화했으므로 메모리 활용도 지표를 모니터링하여 **나중에 인스턴스의 사용률이 지나치게 높아지거나 낮아지지 않도록 하는 것이 중요함**

이를 용이하게 하기 위해 이제 **메모리 사용률을 모니터링하도록 CloudWatch 경고 설정하기**

태스크 7.1: DB-SERVER의 메모리 경고

- CloudWatch > 왼쪽 탐색 창의 **All Alarms** > **Create alarm**
- **Select metric** > **Custom namespaces** 섹션에서 **MemoryUsage** 카드를 선택 > **Instanceld** 카드를 선택 > **DB-Server** 인스턴스를 선택 > **Select metric**을 선택



DB-Server 인스턴스의 메모리 사용률을 보여주는 샘플 그래프가 표시됨

- **Conditions** 섹션
 - **Threshold type**에 **Static** 선택
 - **Whenever mem_used_percent is...** 에 **Greater/Equal** 선택
 - **than...** 에 75를 입력
- **Configure actions** 페이지
 - **Notification** 섹션에서 **Remove**를 선택
- **Add name and description** 페이지
 - **Alarm name**에 **Over-utilized Memory: DB-Server** 를 입력

태스크 7.2: 복합 경고 생성

이전 태스크에서는 Web-Server의 CPU 사용률을 모니터링하는 두 개의 경보를 설정했음
이제 방금 DB-Server에 대해 생성한 메모리 사용률 경고와 이전에 Web-Server에 대해 생성한 CPU 사용률 과다 경보를 결합한 복합 경보를 생성하기

이렇게 하면 단일 경보로 여러 EC2 인스턴스의 성능 문제를 모니터링할 수 있습니다.

- Over Utilized Memory: DB-Server , Over Utilized CPU: Web-Server 선택 > Create composite alarm 선택
- **Conditions** 섹션에는 다음 스테이트먼트가 미리 채워져 있음

CloudWatch > Alarms > Create composite alarm

Step 1
Specify composite alarm conditions

Step 2
Configure actions

Step 3
Add name and description

Step 4
Preview and create composite alarm

Specify composite alarm conditions

Conditions

Composite alarm conditions

This alarm will go in alarm when the following rule is met. Configure by using AND/OR in the text editor. [Info](#)

[Add another alarm](#)

```
1 ALARM("Over-utilized Memory: DB-Server") OR
2 ALARM("Over-utilized CPU: Web-Server")
```

Cancel Next

- **Configure actions** 페이지의 **Notification** 섹션에서 **Remove**를 선택
- **Add name and description** 페이지
 - **Alarm name**에 **Web Application: Over Utilized** 를 입력
- 결과

CloudWatch

Successfully created alarm Web Application: Over Utilized. [View alarm](#)

CloudWatch > Alarms

Alarms (1/4)

Search

Alarm state: Any Alarm type: Any Actions status: Any

Name	State	Last state update	Conditions	Actions
Web Application: Over Utilized	OK	2024-04-02 06:46:19	Any of the alarms goes in Alarm	No actions

CloudWatch > Alarms > Web Application: Over Utilized

Alarms (4)

Search

Alarm state: Any Alarm type: Any Actions status: Any

Hide Auto Scaling alarms

Composite alarm

Over-utilized Memory: DB-Server

Metric alarm

OK

Under-utilized CPU: Web-Server

Metric alarm

In alarm

Over-utilized CPU: Web-Server

Metric alarm

OK

Web Application: Over Utilized

Timeline

This shows the state changes of the composite alarm.

Time range: 1h 3h 12h 1d 3d 1w 2w

Click timeline to see triggering alarms at the selected time.

Child alarms

Search

Name	State	Last state update	Type
Over-utilized Memory: DB-Server	OK	2024-04-02 06:44:57	Metric alarm
Over-utilized CPU: Web-Server	OK	2024-04-02 05:13:57	Metric alarm

Details Tags Alarm rule Actions History Parent alarms

Details

⇒ DB 서버 인스턴스에 대한 메모리 사용률 경보를 성공적으로 생성함