개발자 가이드

AWS DeepRacer

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS DeepRacer: 개발자 가이드

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon의 상표 및 트레이드 드레스는 Amazon 외 제품 또는 서비스와 함께, Amazon 브랜드 이미지를 떨어뜨리거나 고객에게 혼동을 일으킬 수 있는 방식으로 사용할 수 없습니다. Amazon이 소유하지 않은 기타 모든 상표는 Amazon과 제휴 관계이거나 관련이 있거나 후원 관계와 관계없이 해당 소유자의 자산입니다.

Table of Contents

AWS DeepRacer란 무엇입니까? 1
AWS DeepRacer 콘솔 1
AWS DeepRacer 차량 2
AWS DeepRacer 리그 2
강화 학습 탐색 3
개념 및 용어 4
레이싱 이벤트 용어 8
작동 방법 10
강화 학습 10
행동 공간 및 보상 함수 12
훈련 알고리즘 15
AWS DeepRacer 워크플로 16
시뮬레이션 환경과 실제 환경의 성능 차이 17
시작 19
첫 번째 모델 훈련 19
AWS DeepRacer 콘솔을 사용하여 강화 훈련 모델을 훈련하려면 19
모델 이름과 환경을 지정하십시오. 19
레이스 유형과 훈련 알고리즘을 선택하십시오. 20
행동 공간 정의 21
가상 자동차 선택 25
보상 함수를 사용자 지정하십시오. 26
시뮬레이션에서 모델 평가 28
모델 훈련 및 평가 33
레이스 유형 및 이를 가능하게 하는 센서 이해 34
센서 선택 35
훈련을 위한 에이전트 구성 37
타임 트라이얼을 위한 훈련 조정 38
장애물 회피 레이스를 위한 훈련 조정 39
헤드-투-봇 레이스를 위한 훈련 조정 41
AWS DeepRacer 콘솔을 사용하여 모델 훈련 및 평가 42
보상 함수 생성 43
행동 공간 탐색 45
하이퍼파라미터 튜닝 47
훈련 작업 진행 상황 조사 52

훈련된 모델 복제 54
시뮬레이션에서 모델 평가 54
실제 환경에 대한 훈련 최적화 55
보상 함수 참조 58
보상 함수 입력 파라미터 58
보상 함수 예제 73
..... 78
AWS DeepRacer 모델을 Amazon S3에 복사 78
AWS DeepRacer 모델을 콘솔로 가져오기 80
문제 해결 82
차량 작동 85
차량 알아보기 85
차량 검사 86
배터리 충전 및 설치 88
컴퓨팅 모듈 테스트 90
디바이스 전원 끄기 90
LED 표시등 91
디바이스 예비 부품 93
차량 설정 103
Wi-Fi 설정 준비 104
Wi-Fi 설정 및 소프트웨어 업데이트 105
디바이스 콘솔 시작 106
차량 보정 108
모델 업로드 116
차량 주행 117
AWS DeepRacer 차량 수동 주행 117
AWS DeepRacer 차량 자율 주행 118
차량 설정 검사 및 관리 120
차량 로그 보기 126
AWS DeepRacer 디바이스 업데이트 및 복원 128
디바이스 소프트웨어 버전을 확인하십시오. 128
Ubuntu 20.04 설치 미디어 만들기 129
..... 129
사전 조건 129
준비 129
부팅 가능한 USB 드라이브 준비 130

Ubuntu 20.04로 기기 업데이트 139
물리적 트랙 빌드 141
재료와 도구 141
필요할 수 있는 재료 141
필요할 수 있는 도구 142
트랙 설치 142
크기 요건 142
모델 성능 고려 사항 144
트랙 빌드 단계 144
트랙 설계 템플릿 149
A~Z 스피드웨이(기본) 트랙 템플릿 150
AWS DeepRacer 스마일 스피드웨이(중급) 트랙 템플릿 151
RL 스피드웨이(고급) 트랙 템플릿 152
단일 회전 트랙 템플릿 152
S-곡선 트랙 템플릿 153
순환 트랙 템플릿 154
레이스 참가 157
레이싱 이벤트 유형 157
온라인 AWS후원 또는 커뮤니티 후원 레이스 참가 157
가상 서킷 레이스에 참여하십시오. 158
커뮤니티 레이스 참가 159
레이스 참가자로 AWS DeepRacer 커뮤니티 레이스에 참가하려면 160
라이브 레이스에 참여하십시오. 167
레이스 주최 171
..... 171
레이스 빠른 시작 만들기 171
레이스를 사용자 지정하십시오. 175
라이브 레이스를 진행해 보십시오. 182
레이스를 생중계하십시오. 187
주최자 역할 187
방송사 장면 187
AWS DeepRacer 장면 템플릿 188
레이스 관리 193
이벤트 주최 197
AWS DeepRacer 이벤트란 무엇입니까? 197
이벤트의 작동 방식 및 예상 사항 197

시작하기 전에 고려해야 할 사항 198
AWS DeepRacer 레이스의 유형 199
모범 사례 200
이벤트 시작하기 201
AWS DeepRacer 이벤트 예제 201
추가 리소스 205
다중 사용자 모드 206
관리자 설정 206
다중 사용자 이해관계자 207
1단계. AWS DeepRacer 다중 사용자 모드의 필수 조건 207
2단계: 다중 사용자 계정 모드 활성화 209
3단계: 후원을 받을 참가자 초대 210
4단계: 사용량 할당량 설정 210
5단계: 사용량 모니터링 211
다음 단계 213
참가자 설정 213
사전 조건 214
1단계. 후원 계정의 자격 증명을 사용하여 AWS 콘솔에 로그인합니다. 214
2단계. AWS 플레이어 계정 생성 또는 로그인 214
3단계. 프로필 사용자 지정하기 215
4단계. 모델 훈련 216
5단계. 후원 사용량 보기 216
6단계. (선택 사항) 추가 후원 시간 요청 216
교육자 도구 217
AWS DeepRacer Student를 강의실에 통합하십시오 217
학생 커뮤니티 레이스 생성 217
학생 레이스 생성 218
학생 레이스를 사용자 지정합니다. 219
학생 레이스 관리 222
보안 225
데이터 보호 225
AWS DeepRacer 종속 서비스 226
필요 IAM 역할 228
AWS Identity and Access Management 228
대상 229
ID를 통한 인증 229

정책을 사용하여 액세스 관리 ... 232
AWS DeepRacer가 IAM과 작동하는 방법 ... 234
자격 증명 기반 정책 예제 ... 240
AWS 관리형 정책 ... 243
교차 서비스 혼동된 대리인 방지 ... 247
문제 해결 ... 249
태그 지정 ... 253
새 리소스에 대한 태그 추가, 보기, 편집 ... 254
기존 리소스에 대한 태그 추가, 보기 및 편집 ... 255
일반적인 문제 해결 ... 258
일반적인 AWS DeepRacer 라이브 문제를 해결하는 방법 ... 258
라이브 레이스 페이지에서 레이스 비디오를 볼 수 없습니다. ... 258
레이스 대기열에 있는 레이서 이름이 빨간색으로 표시됩니다. ... 259
라이브 레이스를 진행 중인데 레이서를 런칭할 수 없어요 ... 260
Chrome 또는 Firefox 브라우저를 사용하고 있는데 라이브 레이스를 보는 데 여전히 문제가 있습니다. ... 261
내 컴퓨터와 차량 간 USB 연결을 통해 디바이스 콘솔에 연결할 수 없는 이유는 무엇입니까? ... 262
AWS DeepRacer 컴퓨팅 모듈 전원은 배터리에서 콘센트로 어떻게 전환합니까? ... 265
USB 플래시 드라이브를 사용하여 Wi-Fi 네트워크에 AWS DeepRacer을 연결하는 방법 ... 266
차량의 주행 모듈 배터리를 충전하는 방법 ... 271
차량의 컴퓨팅 모듈 배터리를 충전하는 방법 ... 274
배터리가 충전되었지만 차량이 움직이지 않음 ... 275
차량 배터리 잠금 문제 해결 ... 278
차량 배터리 잠금을 방지하는 방법 ... 279
AWS DeepRacer 차량 배터리를 잠금 해제하는 방법 ... 279
LiDAR 센서를 설치할 때 Dell 배터리 커넥터 케이블을 감싸는 방법 ... 282
차량의 연결을 유지하는 방법 ... 287
차량의 Wi-Fi LED 표시기가 파란색으로 깜박인 다음 2초 동안 빨간색으로 바뀐 뒤 마지막으로 꺼진 경우 Wi-Fi 연결 문제를 해결하는 방법 ... 287
차량의 Wi-Fi 또는 전원 LED 표시기가 파란색으로 깜박이면 무슨 의미입니까? ... 288
어떻게 하면 호스트 이름을 사용하여 차량의 디바이스 콘솔에 연결할 수 있습니까? ... 289
IP 주소를 사용하여 차량의 디바이스 콘솔에 연결하는 방법 ... 289
디바이스의 Mac 주소를 가져오는 방법 ... 289
디바이스 콘솔의 기본 암호를 복구하는 방법 ... 290
디바이스를 수동으로 업데이트하는 방법 ... 291
일반적인 디바이스 운영 문제를 진단하고 해결하는 방법 ... 292

디바이스 콘솔의 비디오 플레이어에 내 차량 카메라의 비디오 스트림이 표시되지 않는 이유는 무엇입니까? 293
내 AWS DeepRacer 차량이 움직이지 않는 이유는 무엇입니까? 293
최신 디바이스 업데이트가 표시되지 않는 이유는 무엇입니까? 최신 업데이트를 받으려면 어떻게 해야 합니까? 294
내 AWS DeepRacer 차량이 내 Wi-Fi 네트워크에 연결되지 않는 이유는 무엇입니까? 294
AWS DeepRacer 디바이스 콘솔 페이지의 로딩 시간이 오래 걸리는 이유는 무엇입니까? 294
AWS DeepRacer 차량에 배포할 때 모델이 제대로 작동하지 않는 이유는 무엇입니까? 295
문서 기록 296
AWS 용어집 299
..... ccc

AWS DeepRacer란 무엇입니까?

AWS DeepRacer는 강화 학습으로 구동되는 완전 자율 주행 1/18 비율의 경주용 자동차입니다. 이 학습 시스템의 구성 요소는 다음과 같습니다.

- AWS DeepRacer 콘솔: 3차원 시뮬레이션 자율 주행 환경에서 강화 학습 모델을 훈련 및 평가하기 위한 AWS 기계 학습 서비스입니다.
- AWS DeepRacer 차량: 자율 주행을 목적으로 훈련된 AWS DeepRacer 모델에 대해 추론을 실행할 수 있는 1/18 비율의 RC 자동차
- AWS DeepRacer 리그: 누구나 참가할 수 있는 세계 최초의 글로벌 자율 주행 레이싱 리그. 상품, 명예, 그리고 World Championship Cup에 진출할 수 있는 기회를 얻을 수 있는 레이스. 자세한 내용은 용어 및 조건 단원을 참조하십시오.

주제

- AWS DeepRacer 콘솔
- AWS DeepRacer 차량
- AWS DeepRacer 리그
- AWS DeepRacer를 사용하여 강화 학습을 탐색합니다.
- AWS DeepRacer 개념 및 용어

AWS DeepRacer 콘솔

AWS DeepRacer 콘솔은 AWS DeepRacer 서비스와 상호 작용할 수 있는 그래픽 사용자 인터페이스입니다. 콘솔을 사용하면 강화 학습 모델을 훈련하거나, AWS DeepRacer 시뮬레이터에서 모델 성능을 평가할 수 있습니다. 또한 물리적 환경에서 자율 주행할 목적으로 훈련된 모델을 다운로드하여 AWS DeepRacer 차량에 배포할 수도 있습니다.

AWS DeepRacer 콘솔은 다음 기능을 지원합니다.

- 훈련 작업을 생성한 후 지정된 보상 함수, 최적화 알고리즘, 환경 및 하이퍼파라미터를 사용하여 강화 학습 모델을 훈련할 수 있습니다.
- SageMaker AI를 사용하여 모델을 훈련하고 평가할 시뮬레이션된 트랙을 선택합니다.
- 훈련된 모델을 복제한 후 하이퍼파라미터 튜닝으로 훈련을 개선하여 모델 성능을 최적화할 수 있습니다.

- 물리적 환경에서 주행할 목적으로 훈련된 모델을 다운로드하여 AWS DeepRacer 차량에 배포할 수 있습니다.
- 모델을 가상 레이스에 제출한 후 다른 모델과 비교하여 성능 순위를 가상 리더보드에 올릴 수 있습니다.

AWS DeepRacer 서비스 콘솔을 사용하는 경우 모델을 훈련 또는 평가 및 저장하는 사용량을 기준으로 요금이 부과됩니다.

시작하는 데 도움이 되도록 AWS DeepRacer는 AWS DeepRacer를 처음 사용하는 사용자에게 프리 티어를 제공합니다. 첫 모델을 훈련 및 조정하고 AWS DeepRacer 리그에 참가하는 데 충분한 시간입니다. AWS DeepRacer 리그 가상 이벤트에 참여하기 위해 모델을 제출하는 경우 요금이 부과되지 않습니다.

요금에 대한 자세한 내용은 AWS DeepRacer 서비스 세부 정보 페이지를 참조하십시오.

AWS DeepRacer 차량

AWS DeepRacer 차량은 Wi-Fi가 지원되는 물리적 차량으로서 물리적 트랙에서 강화 학습 모델을 사용해 스스로 주행할 수 있습니다.

- 수동으로 차량을 제어하거나, 혹은 자율 주행을 목적으로 모델을 차량에 배포할 수도 있습니다.
- 자율 주행 모드일 때는 차량의 컴퓨팅 모듈에서 추론을 실행합니다. 추론은 차량 전면에 장착된 카메라에서 수집되는 이미지를 사용합니다.
- Wi-Fi가 연결되어 차량에서 소프트웨어를 다운로드할 수 있습니다. 또한 사용자가 컴퓨터 또는 모바일 디바이스를 사용해 디바이스 콘솔에 액세스하여 차량을 구동하는 것도 가능합니다.

AWS DeepRacer 리그

AWS DeepRacer 리그는 AWS DeepRacer의 중요한 구성 요소입니다. AWS DeepRacer 리그는 커뮤니티와 경쟁을 촉진하기 위한 것입니다.

AWS DeepRacer League를 사용하면 실제 또는 가상 레이싱 이벤트에서 ML 기술을 다른 AWS DeepRacer 개발자와 비교할 수 있습니다. 상금과 성과를 얻을 기회뿐만 아니라 강화 학습 모델을 평가하는 방법도 될 것입니다. 다른 참가자와 경쟁하고, 서로 배우고, 영감을 줄 수 있습니다. AWS DeepRacer 리그에서 성과를 거두면 소셜 미디어를 통해 커뮤니티와 공유할 수 있습니다. 자세한 내용은 용어 및 조건 단원을 참조하십시오.

레이스에 참가하거나 리그에서 모델을 훈련하는 방법을 알아보십시오.

AWS DeepRacer를 사용하여 강화 학습을 탐색합니다.

강화 학습, 특히 심층 강화 학습은 광범위한 자율 의사결정 문제를 해결하는 데 그 효율성을 이미 입증했으며, 두 세 가지 예만 들자면, 금융 거래, 데이터 센터 냉각, 병참 및 자율 레이싱 등에서 응용되고 있습니다.

강화 학습은 실제 문제를 해결할 수 있는 잠재력을 가지고 있습니다. 하지만 광범위한 기술 범위와 전문성으로 인해 학습 곡선이 가파릅니다. 실제 실험을 진행하려면 자율 주행 레이싱 자동차와 같은 물리적 에이전트가 필요합니다. 또한 주행 트랙 또는 공공도로와 같은 물리적 환경도 확보해야 합니다. 이러한 환경은 비용과 시간이 많이 들고, 위험할 수 있습니다. 단순히 강화 학습을 이해하는 것만으로는 이러한 요건들을 해결하지 못합니다.

학습 곡선을 완만하게 만들기 위해 AWS DeepRacer에서는 다음 3가지 방법으로 프로세스를 간소화했습니다.

- 강화 학습 모델을 훈련하고 평가할 때 단계별 지침을 제공합니다. 이 지침에는 사전 설정된 환경, 상태 및 행동과 사용자 지정이 가능한 보상 함수가 포함되어 있습니다.
- 가상 에이전트와 가상 환경 간 상호 작용을 에뮬레이션하는 시뮬레이터를 제공합니다.
- AWS DeepRacer 차량을 물리적 에이전트로 사용. 이 차량을 사용해 훈련된 모델을 물리적 환경에서 평가할 수 있습니다. 이는 실제 사용 사례와 매우 유사합니다.

숙련된 기계 학습 전문가들에게 AWS DeepRacer는 가상 환경과 물리적 환경에서 자율 레이싱을 위한 강화 학습 모델을 개발할 수 있는 통합 시스템으로 손색이 없습니다. 요컨대, AWS DeepRacer를 사용해 다음 단계에 따라 자율 주행이 가능한 강화 학습 모델을 개발할 수 있습니다.

1. 자율 레이싱을 위한 맞춤형 강화 학습 모델을 훈련합니다. SageMaker AI와 통합된 AWS DeepRacer 콘솔을 사용하여이 작업을 수행합니다. SageMaker
2. AWS DeepRacer 시뮬레이터를 사용해 가상 환경에서 모델을 평가하고 자율 주행 레이싱을 테스트합니다.
3. 훈련된 모델을 AWS DeepRacer 모형 차량에 배포하여 물리적 환경에서 자율 주행을 테스트합니다.

AWS DeepRacer 개념 및 용어

AWS DeepRacer는 다음 개념을 토대로 빌드되었으며 다음 용어를 사용합니다.

AWS DeepRacer 서비스

AWS DeepRacer는 자율 레이싱에 중점을 둔 강화 학습을 살펴보기 위한 AWS 기계 학습 서비스입니다. AWS DeepRacer 서비스는 다음 기능을 지원합니다.

1. 클라우드에서 강화 학습 모델을 훈련합니다.
2. AWS DeepRacer 콘솔에서 훈련된 모델을 평가합니다.
3. 훈련된 모델을 가상 레이스에 제출하고, 적합한 경우 성능을 이벤트의 리더보드에 게시합니다.
4. 성능 개선을 위해 훈련된 모델을 복제하고 훈련을 지속합니다.
5. AWS DeepRacer 차량에 업로드할 수 있도록 훈련된 모델 아티팩트를 다운로드합니다.
6. 차량을 자율 주행을 위한 물리적 트랙에 배치하고 모델의 실제 성능을 평가합니다.
7. 필요하지 않은 모델을 삭제하여 불필요한 요금이 부과되지 않도록 합니다.

AWS DeepRacer

“AWS DeepRacer”는 다음의 세 가지 차량을 의미할 수 있습니다.

- 가상 레이싱 자동차는 오리지널 AWS DeepRacer 디바이스, Evo 디바이스 또는 AWS DeepRacer 리그 가상 서킷 레이스에 참가하여 획득할 수 있는 다양한 디지털 보상의 형태를 취할 수 있습니다. 색상을 변경하여 가상 자동차를 사용자 지정할 수도 있습니다.
- 오리지널 AWS DeepRacer 디바이스는 1/18 비율의 실제 모델 자동차입니다. 이 차량에는 카메라와 온보드 컴퓨팅 모듈이 장착되어 있습니다. 컴퓨팅 모듈은 트랙을 따라 스스로 주행하기 위해 추론을 실행합니다. 컴퓨팅 모듈과 차량 섀시는 각각 컴퓨팅 배터리와 드라이브 배터리라고 하는 전용 배터리로 구동됩니다.
- AWS DeepRacer Evo 디바이스는 옵션 센서 키트를 가진 오리지널 디바이스입니다. 이 키트에는 추가 카메라와 LIDAR(조명 감지 및 거리 측정)가 포함되어 있어 차량이 후방 및 측면 물체를 감지할 수 있습니다. 이 키트에는 새 쉘도 포함되어 있습니다.

강화 학습

강화 학습은 환경과의 상호 작용을 통해 지정된 목표를 달성할 수 있도록 에이전트의 자율 의사 결정에 중점을 둔 기계 학습 방법입니다. 강화 학습에서는 시행착오를 통해 학습합니다. 훈련에는 레이블이 지정된 입력이 필요하지 않지만, 훈련은 보상 가설에 의존합니다. 이는 작업 시퀀스 이후 미래의 보상을 극대화하여 모든 목표를 달성할 수 있다고 가정합니다. 강화 학습에서는 보상 함수 설계가 중요합니다. 높은 수준의 보상 함수를 만들 경우 에이전트가 더 나은 결정을 내릴 수 있게 합니다.

자율 레이싱의 경우 에이전트는 차량입니다. 환경에는 이동 경로와 교통 조건이 포함됩니다. 목표는 차량이 사고 없이 신속하게 목표 지점에 도착하는 것입니다. 보상은 목표 지점까지 안전하고 빠른 주행을 촉진하기 위해 사용되는 점수입니다. 위험하거나 비생산적인 주행에는 벌점이 부과됩니다.

훈련 중 학습을 장려하기 위해 학습 중인 에이전트는 경우에 따라 보상이 제공되지 않을 수도 있는 작업을 추구하도록 허용해야 합니다. 이를 탐색과 활용의 상충 관계라고 합니다. 이는 에이전트가 잘못된 목표 지점으로 잘못 안내될 가능성을 줄이거나 없애줍니다.

보다 자세한 공식적인 정의는 Wikipedia에서 강화 학습을 참조하십시오.

강화 학습 모델

강화 학습 모델은 에이전트가 행동하는 환경에서는 에이전트의 상태, 에이전트가 수행할 수 있는 작업 및 작업 수행으로 인해 받는 보상의 세 가지 사항을 설정합니다. 에이전트가 작업을 결정하는 데 사용하는 전략을 정책이라고 합니다. 정책은 환경 상태를 입력으로 사용하여 수행할 작업을 출력합니다. 강화 학습에서 정책은 종종 심층 신경망으로 표현됩니다. 이를 강화 학습 모델이라고 합니다. 훈련 작업마다 모델이 하나씩 생성됩니다. 훈련 작업이 조기에 중지되는 경우에도 모델을 생성할 수 있습니다. 모델은 변경이 불가능합니다. 즉, 생성한 뒤에는 수정 및 덮어 쓸 수 없습니다.

AWS DeepRacer 시뮬레이터

AWS DeepRacer 시뮬레이터는 AWS DeepRacer 모델의 훈련 및 평가를 시각화하기 위한 가상 환경입니다.

AWS DeepRacer 차량

AWS DeepRacer를 참조하십시오.

AWS DeepRacer 자동차

이 유형의 AWS DeepRacer 차량은 1/18 비율의 모델 자동차입니다.

리더보드

리더보드는 AWS DeepRacer 리그 레이싱 이벤트에서 AWS DeepRacer 차량 성능의 순위를 매긴 목록입니다. 여기에서 레이싱이란 시뮬레이션 환경에서 실행하는 가상 이벤트이거나 실제 환경에서 실행하는 물리적 이벤트일 수 있습니다. 성능 지표는 레이싱 유형에 따라 달라집니다. 주어진 레이스 트랙과 동일하거나 유사한 트랙에서 훈련된 모델을 평가한 AWS DeepRacer 사용자가 제출한 가장 빠른 랩 타임, 총 시간 또는 평균 랩 타임일 수 있습니다.

차량이 연속해서 3바퀴를 완주하면 리더보드에 순위가 매겨집니다. 첫 연속 3바퀴의 평균 랩 타임이 리더보드에 제출됩니다.

기계 학습 프레임워크

기계 학습 프레임워크는 기계 학습 알고리즘을 구축하는 데 사용된 소프트웨어 라이브러리입니다. AWS DeepRacer에 대해 지원되는 프레임워크로는 Tensorflow가 있습니다.

정책 네트워크

정책 네트워크는 훈련된 신경망으로, 비디오 이미지를 입력으로 사용하여 에이전트의 다음 작업을 예측합니다. 알고리즘에 따라 에이전트의 현재 상태 값을 평가할 수도 있습니다.

최적화 알고리즘

최적화 알고리즘은 모델을 훈련시키는 데 사용되는 알고리즘입니다. 지도식 훈련의 경우 이 알고리즘은 가중치 업데이트를 위한 특정 전략을 사용하여 손실 함수를 최소화함으로써 최적화됩니다. 강화 학습의 경우에는 특정 보상 함수를 사용하여 미래의 보상 기대치를 극대화함으로써 최적화됩니다.

신경망

신경망(인공 신경망이라고도 함)은 생물학적 시스템을 기반으로 정보 모델을 구축하는 데 사용되는 연결된 유닛 또는 노드의 집합입니다. 각 노드는 생체 뉴런을 모방한 인공 뉴런으로 불리며, 여기에서 입력 신호(자극)를 수신합니다. 이때 입력 신호가 충분히(활성화할 정도로) 강력할 경우 노드가 활성화되어 입력 신호와 활성화를 근거로 예상되는 출력을 생성합니다. 신경망은 기계 학습에서 널리 사용됩니다. 인공 신경망은 어떤 함수든지 범용 근사의 역할을 할 수 있기 때문입니다. 기계가 학습을 하면 입력값과 출력값에 대해 최적의 함수 근사값을 찾을 수 있습니다. 심층 강화 학습에서는 신경망이 정책을 의미하여 종종 정책망이라고도 불립니다. 정책망 훈련은 현재 정책을 기반으로 여러 단계에 걸쳐 경험 생성을 반복한 후 새롭게 생성된 경험으로 정책망을 최적화하는 프로세스입니다. 이러한 프로세스는 일정한 성능 지표가 필요한 기준으로 충족할 때까지 계속 됩니다.

하이퍼파라미터

하이퍼파라미터는 신경망 훈련의 성능을 제어하는 알고리즘 기반 변수입니다. 하이퍼파라미터의 한 예로 각 단계에서 학습할 때 얼마나 많은 새로운 경험이 계산되는지를 제어하는 학습 속도가 있습니다. 학습 속도가 빠를수록 훈련 속도도 빠르지만 훈련된 모델의 품질이 떨어질 수도 있습니다. 하이퍼파라미터는 실증적이지만 각 훈련마다 체계적인 튜닝이 필요합니다.

AWS DeepRacer 트랙

트랙은 AWS DeepRacer 차량이 주행하는 경로 또는 코스입니다. 트랙은 시뮬레이션 환경 또는 실제 물리적 환경에 존재할 수 있습니다. 시뮬레이션 환경은 AWS DeepRacer 모델을 가상 트랙에서 훈련하는 데 사용됩니다. AWS DeepRacer 콘솔에서는 가상 트랙을 사용할 수 있습니다. 실제 환경은 AWS DeepRacer 차량을 물리적 트랙에서 구동하는 데 사용됩니다. AWS DeepRacer 리

그는 이벤트 참가자들이 경쟁할 수 있는 물리적 트랙을 제공합니다. 다른 상황에서 자신의 AWS DeepRacer 차량을 구동하려면 물리적 트랙을 만들어야 합니다. 트랙을 구축하는 방법에 대해 자세히 알아보려면 물리적 트랙 구축을 참조하십시오.

보상 함수

보상 함수는 에이전트에게 다음과 같은 수행한 작업의 결과를 알려주는 학습 모델 내 알고리즘입니다.

- 강화해야 하는 좋은 결과
- 보통 결과
- 권장되지 않는 안 좋은 결과

보상 함수는 강화 학습의 핵심 부분으로, 다른 작업보다 특정 작업에 대해 인센티브를 제공해 에이전트가 학습할 동작을 결정합니다. 사용자는 Python을 사용하여 보상 함수를 제공합니다. 최적화 알고리즘에서는 이 보상 함수를 사용하여 강화 학습 모델을 훈련합니다.

경험 에피소드

경험 에피소드는 에이전트가 지정된 출발점에서 시작하여 트랙을 완주하거나 트랙에서 벗어날 때까지 실행하면서 환경에서 데이터를 훈련하여 얻어진 경험을 수집하는 기간입니다. 에피소드에 따라 길이가 달라질 수 있습니다. 이를 에피소드 또는 경험 생성 에피소드라고 합니다.

경험 반복

경험 생성 반복이라고도 알려진 경험 반복은 정책망 가중치를 업데이트하는 각 정책 반복 사이에서 연속되는 경험의 집합을 말합니다. 각 경험 반복의 마지막에서 수집된 에피소드가 경험 다시 재생 또는 버퍼에 추가됩니다. 크기는 훈련 하이퍼파라미터 중 하나에서 설정할 수 있습니다. 신경망은 임의의 경험 샘플을 사용하여 업데이트됩니다.

정책 반복

정책 업데이트 반복이라고도 알려진 정책 반복은 경사 상승 과정에서 정책 신경망 가중치를 업데이트하기 위해 무작위로 샘플링되는 훈련 데이터를 통과한 패스(pass)들로 구성됩니다. 가중치 업데이트를 목적으로 훈련 데이터를 통과하는 단일 패스를 에포크라고 부르기도 합니다.

훈련 작업

훈련 작업은 강화 학습 모델을 훈련하고 추론을 실행할 훈련된 모델 아티팩트를 생성하는 워크로드입니다. 각 훈련 작업에는 두 개의 하위 프로세스가 있습니다.

1. 에이전트를 시작하고 현재 정책을 따릅니다. 에이전트가 다양한 에피소드에서 환경을 탐색하고 훈련 데이터를 생성합니다. 이러한 데이터 생성이 반복 프로세스입니다.

2. 새로운 훈련 데이터를 적용하여 새 정책 경사를 컴퓨팅합니다. 네트워크 가중치를 업데이트하고 계속 훈련합니다. 중지 조건이 충족될 때까지 1단계를 반복합니다.

각 훈련 작업마다 훈련된 모델을 생성하여 모델 아티팩트를 지정된 데이터 스토어로 출력합니다.

평가 작업

평가 작업은 모델의 성능을 테스트하는 워크로드입니다. 훈련 작업이 완료되면 지정된 지표에 따라 성능이 측정됩니다. 표준 AWS DeepRacer 성능 지표는 에이전트가 트랙에서 랩을 완주하는 데 걸린 주행 시간입니다. 다른 지표는 완주한 랩의 비율입니다.

레이싱 이벤트 용어

AWS DeepRacer 레이싱 이벤트는 다음 개념과 용어를 사용합니다.

리그/대회

AWS DeepRacer League 이벤트의 맥락에서 리그 및 경쟁이라는 용어는 경쟁 구조와 관련이 있습니다. AWS DeepRacer 리그는 AWS DeepRacer 리그 AWS 를 후원합니다. 즉, AWS DeepRacer 리그를 소유하고 설계하고 실행합니다. 대회에는 시작 날짜와 종료 날짜가 있습니다.

Season

대회는 몇 년간 반복해서 열릴 수 있습니다. 이를 시즌(예: 2019 시즌 또는 2020 시즌)이라고 부릅니다. 규칙은 시즌에 따라 변경될 수 있지만 일반적으로 한 시즌 동안은 일관되게 유지됩니다. AWS DeepRacer 리그 이용 약관은 시즌에 따라 달라질 수 있습니다.

가상 서킷

가상 서킷은 AWS DeepRacer 리그 시즌 동안 AWS DeepRacer 콘솔에서 AWS 열리는 레이스를 말합니다.

Event

규칙의 정의에 따르면 이벤트는 레이스에 참가할 수 있는 AWS DeepRacer 리그 행사입니다. 이벤트에는 시작 날짜와 종료 날짜가 있습니다. 가상 서킷 이벤트는 일반적으로 한 달 동안 진행됩니다. 한 시즌에는 많은 이벤트가 있을 수 있으며, 이벤트 참가자의 순위를 매기는 방법, 우승자를 선정하는 방법, 그 후에 일어나는 일 등과 같은 일부 규칙이 변경될 수 있습니다.

레이스 유형

모든 레이서는 타임 트라이얼(TT), 물체 회피(OA) 또는 헤드-투-봇(H2B) 레이스로 레이스를 펼칠 수 있습니다. 각 레이스 유형에 따라 바퀴 수, 레이서 순위 결정 방법이 지정됩니다.

전국 시즌 순위

전국 시즌 순위는 해당 국가의 레이서들 중에서 한 레이서의 순위표 순위를 의미합니다. 모든 레이서는 매월 진행되는 가상 레이스에서 자국의 다른 레이서들과 경쟁할 수 있습니다.

지역 시즌 순위

지역 시즌 순위는 해당 지역의 레이서들 중에서 한 레이서의 순위표 순위를 말합니다.

월드 챔피언십

AWS DeepRacer 리그의 가상 서킷 월간 순위표는 국가 및 지역으로 구분됩니다. 각 리전의 상위 레이서는 AWS re:Invent에서 월드 챔피언십에 참가할 수 있는 자격을 갖추게 됩니다. 자세한 내용은 용어 및 조건 단원을 참조하십시오.

AWS DeepRacer 작동 방식

AWS DeepRacer 차량은 자율적으로 트랙을 따라 주행하거나 다른 차량과 경쟁할 수 있는 1/18 축적 차량입니다. 차량은 전방 카메라, 스테레오 카메라, 레이더(LiDAR) 등 다양한 센서를 장착할 수 있습니다. 센서는 차량이 작동하는 환경에 대한 데이터를 수집합니다. 각 센서는 서로 다른 축적으로 뷰를 제공합니다.

AWS DeepRacer는 강화 학습을 사용하여 AWS DeepRacer 차량의 자율 주행을 구현합니다. 이를 위해 먼저 가상 환경인 시뮬레이션 트랙에서 강화 학습 모델을 훈련하고 평가합니다. 훈련을 마치면 훈련된 모델 아티팩트를 AWS DeepRacer 차량에 업로드합니다. 그런 다음 실제 트랙인 물리적 환경에서 자율 주행할 수 있도록 차량을 설정합니다.

강화 학습 모델을 교육하는 것은 어려울 수 있습니다. 특히 이 분야를 처음 접하는 경우에는 더욱 그렇습니다. AWS DeepRacer는 필수 구성 요소를 통합하고 따라하기 쉬운 마법사와 같은 작업 템플릿을 제공하여 프로세스를 간소화합니다. 그렇더라도 AWS DeepRacer에 구현된 강화 학습 훈련에 대해 기본 정보를 파악해두는 것이 좋습니다.

주제

- AWS DeepRacer에서의 강화 학습
- AWS DeepRacer 행동 공간 및 보상 함수
- AWS DeepRacer 훈련 알고리즘
- AWS DeepRacer 솔루션 워크플로
- 시뮬레이션 환경과 실제 환경의 성능 차이

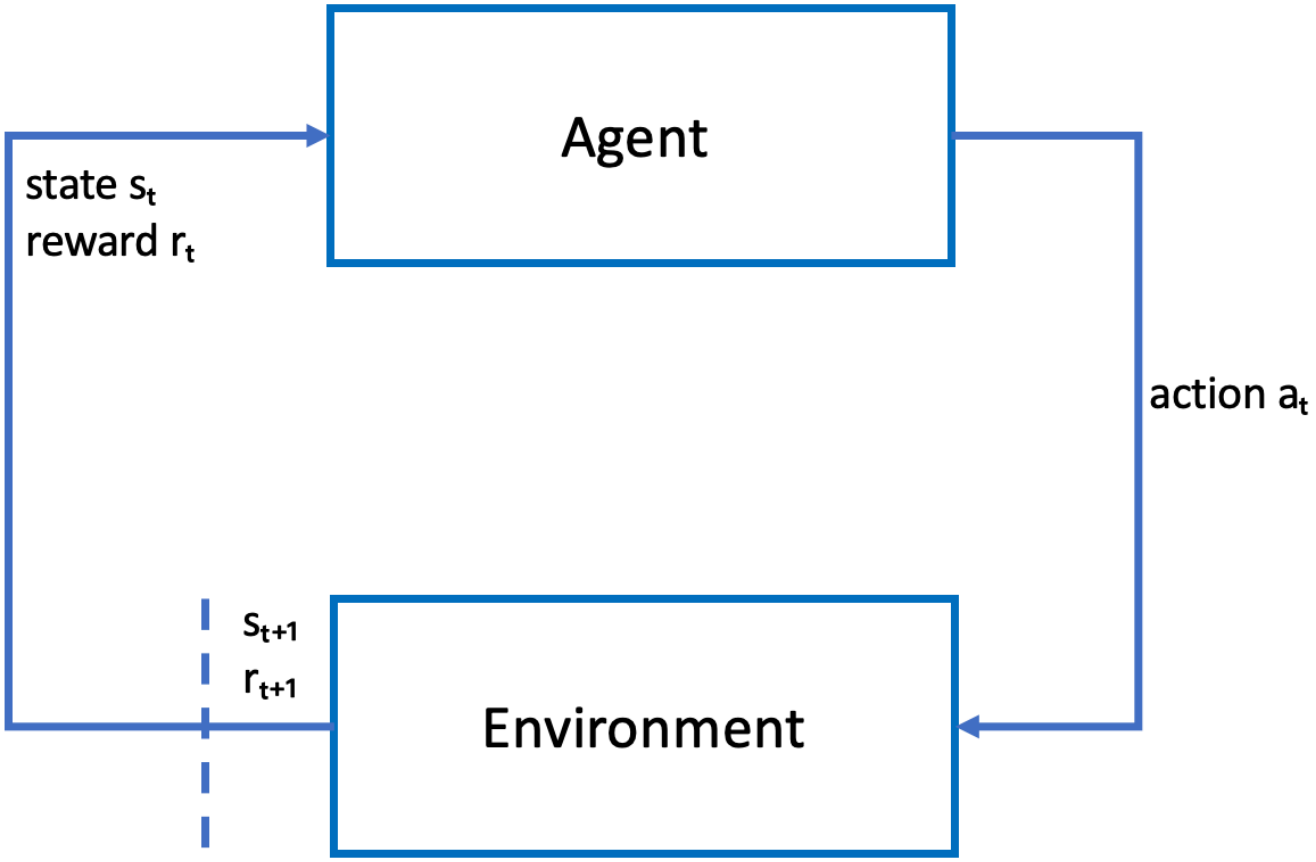
AWS DeepRacer에서의 강화 학습

강화 학습에서는 의도한 목표를 달성하기 위한 목표를 가진 물리적 또는 가상 AWS DeepRacer 차량과 같은 에이전트가 환경과 상호 작용하여 에이전트의 총 보상을 극대화합니다. 에이전트는 정책이라고 하는 전략에 따라 주어진 환경 상태에서 조치를 취하고 새로운 상태에 도달합니다. 어떤 행동이든 즉각적인 보상으로 이어집니다. 이러한 보상은 행동의 적합성을 나타내는 척도입니다. 즉각적인 보상은 환경에서 반환되는 것으로 알려져 있습니다.

AWS DeepRacer에서 강화 학습의 목적은 임의 환경에서 최적의 정책을 학습하는 데 있습니다. 여기에서 학습이란 시행 착오가 반복되는 프로세스를 말합니다. 에이전트는 무작위로 초기 행동을 보이면서 새로운 상태에 도달합니다. 그런 다음 해당 단계를 반복하여 새로운 상태에서 다음 상태로 넘어갑니다.

에이전트는 시간이 지나면서 이러한 방식으로 장기적 보상을 극대화하는 행동을 발견합니다. 초기 상태에서 최종 상태까지 이어지는 에이전트의 상호 작용을 에피소드라고 부릅니다.

다음은 학습 프로세스를 도식으로 나타낸 그림입니다.



에이전트는 신경망을 구체화하고, 신경망은 에이전트의 정책에 대한 근사 함수를 표현합니다. 차량의 정방 카메라에서 촬영된 이미지는 환경 상태이고, 에이전트 행동은 에이전트의 속도와 조향 각도로 결정됩니다.

에이전트가 트랙을 벗어나지 않고 완주하면 양의 보상을, 그리고 트랙에서 벗어나면 음의 보상을 받습니다. 에피소드는 레이스 트랙의 임의 구간에서 에이전트와 함께 시작되어 에이전트가 트랙에서 벗어나거나 한 바퀴를 완주할 때 종료됩니다.

> **Note**
>
> 엄밀히 말해서 환경 상태는 문제와 관련된 모든 것을 말합니다. 예를 들어 트랙의 차량 위치나 트랙 형상도 환경 상태가 될 수 있습니다. 차량 전면에 장착된 카메라를 통해 수집되는 이미지

로는 전체 환경 상태를 알 수 없습니다. 이러한 이유로 환경은 부분적으로 관측된 결과라고 할 수 있으며, 에이전트에 대한 입력 데이터가 상태가 아닌 관측 결과라고 불리는 이유도 여기에 있습니다. 하지만 이번 문서에서는 단순화할 목적으로 상태와 관측 결과를 동일한 의미로 사용합니다.

시뮬레이션 환경에서 에이전트를 훈련하면 다음과 같은 이점이 있습니다.

- 시뮬레이션에서는 에이전트의 진행 상황을 예상하면서 트랙에서 이탈하는 시점을 찾아내 보상을 계산할 수 있습니다.
- 시뮬레이션에서는 마치 물리적 환경에서 훈련하는 것처럼 트랙에서 이탈할 때마다 번거롭게 차량을 재설정할 필요가 없습니다.
- 시뮬레이션은 훈련 속도를 높일 수 있습니다.
- 시뮬레이션에서는 다른 트랙, 배경 및 차량 조건을 선택하는 등 환경 조건을 효과적으로 제어할 수 있습니다.

강화 학습에 대한 대안으로 지도 학습이 있으며, 모방 학습으로도 불립니다. 모방 학습에서는 임의 환경에서 수집되어 알려진 데이터 세트([이미지, 행동] 튜플)가 에이전트 훈련에 사용됩니다. 모방 학습을 통해 훈련되는 모델도 자율 주행에 적용할 수 있습니다. 단, 카메라에서 촬영된 이미지가 훈련 데이터 세트의 이미지와 비슷할 때만 효과가 있습니다. 따라서 견고한 주행을 위해서는 훈련 데이터 세트가 포괄적이어야 합니다. 이와 반대로 강화 학습에서는 포괄적인 라벨링 작업 없이 시뮬레이션에서 완전한 훈련이 가능합니다. 강화 학습은 무작위 행동으로 시작되기 때문에 에이전트가 다양한 환경 및 트랙 조건을 학습합니다. 강화 학습으로 훈련된 모델이 견고한 이유도 여기에 있습니다.

AWS DeepRacer 행동 공간 및 보상 함수

행동 공간

강화 학습에서는 에이전트가 환경과 상호 작용할 때 사용할 수 있는 모든 유효한 행동 또는 선택의 집합을 행동 공간이라고 합니다. AWS DeepRacer 콘솔에서는 개별 또는 연속 작업 공간에서 에이전트를 훈련할 수 있습니다.

불연속 행동 공간

불연속 행동 공간은 유한한 집합의 각 상태에 대해 에이전트가 수행할 수 있는 모든 작업을 나타냅니다. AWS DeepRacer의 경우 이는 모든 환경 상황에 대해 에이전트의 신경망이 카메라 및 (선택 사항)

LiDAR 센서의 입력을 기반으로 차량의 속도와 방향을 선택한다는 것을 의미합니다. 선택은 사전 정의된 조향 각도와 제한 값 조합의 그룹화로 제한됩니다.

AWS DeepRacer 차량은 회전이 가까워지면 가속 또는 제동 후 좌회전, 우회전 또는 직진 중 하나를 선택할 수 있습니다. 이러한 동작은 조향 각도와 속도의 조합으로 정의되어 에이전트를 위한 옵션 메뉴(0~9)를 생성합니다. 예를 들어 0은 -30도 및 0.4m/s, 1은 -30도 및 0.8m/s, 2는 -15도 및 0.4m/s, 3은 -15도 및 0.8m/s, 3은 -15도 및 0.8m/s 등 9까지 나타낼 수 있습니다. 각도가 음수이면 차를 오른쪽으로 돌리고, 양수이면 차를 왼쪽으로 돌리고, 0이면 바퀴가 똑바로 유지됩니다.

AWS DeepRacer의 기본 개별 행동 공간에는 다음과 같은 행동이 포함됩니다.

AWS DeepRacer의 기본 개별 행동 공간

행동 번호	조향 각도	속도
0	-30도	0.4m/s
1	-30도	0.8m/s
2	-15도	0.4m/s
3	-15도	0.8m/s
4	0도	0.4m/s
5	0도	0.8m/s
6	15도	0.4m/s
7	15도	0.8m/s
8	30도	0.4m/s
9	30도	0.8m/s

연속 행동 공간

연속 행동 공간을 통해 에이전트는 각 상태에 대한 값 범위에서 행동을 선택할 수 있습니다. 개별 행동 공간과 마찬가지로, 이는 환경 상황이 점점 달라질 때마다 에이전트의 신경망이 카메라와(선택 사항) LiDAR 센서의 입력을 기반으로 차량의 속도와 방향을 선택한다는 것을 의미합니다. 하지만 연속 행동 공간에서는 에이전트가 액션을 선택하는 옵션 범위를 정의할 수 있습니다.

이 예제에서 회전이 가까워지는 연속 동작 공간에 있는 AWS DeepRacer 차량은 0.75m/s에서 4m/s 사이의 속도를 선택하고 -20도에서 20도 사이의 조향 각도를 선택하여 좌회전, 우회전 또는 직진할 수 있습니다.

불연속형 및 연속형

연속 행동 공간을 사용하면 트랙의 특정 지점에서의 속도/조향 동작을 장려하여 성능을 최적화하는 보상 함수를 작성할 수 있다는 이점이 있습니다. 또한 다양한 동작 중에서 선택하면 속도 및 조향 값을 부드럽게 변경할 수 있으므로 잘 훈련된 모델에서는 실제 상황에서 더 나은 결과를 얻을 수 있습니다.

불연속 작업 공간 설정에서 에이전트의 선택을 미리 정의된 유한한 수의 행동으로 제한하면 이러한 행동이 미치는 영향을 이해하고 환경(트랙, 레이싱 형식) 및 보상 함수를 기반으로 행동을 정의해야 하는 부담이 있습니다. 하지만 연속 작업 공간 설정에서는 에이전트가 훈련을 통해 제공한 최소/최대 범위 중에서 최적의 속도 및 조향 값을 선택하는 방법을 학습합니다.

모델에서 선택할 수 있는 다양한 값을 제공하는 것이 더 나은 방법인 것 같지만 에이전트는 최적의 동작을 선택하는 방법을 배우려면 더 오래 훈련해야 합니다. 성공 여부는 보상 함수 정의에 따라서도 좌우됩니다.

보상 함수

에이전트는 환경을 탐색하면서 가치 함수를 학습합니다. 가치 함수는 에이전트가 환경을 관찰한 후 취한 행동이 얼마나 좋은지 판단하는 데 도움이 됩니다. 가치 함수는 AWS DeepRacer 콘솔에서 작성한 보상 함수를 사용하여 행동에 점수를 매깁니다. 예를 들어, AWS DeepRacer 콘솔의 센터 라인 따르기 샘플 보상 기능에서 좋은 행동은 에이전트를 트랙 중앙에 가깝게 유지하고 에이전트를 트랙 중앙에서 멀어지게 하는 잘못된 행동보다 높은 점수를 받게 하는 것입니다.

시간이 지남에 따라 에이전트는 가치 함수를 통해 총 보상을 늘리는 정책을 학습할 수 있습니다. 최적 또는 최선의 정책은 에이전트가 환경을 탐색하는 데 소비하는 시간과 경험을 통해 정책을 학습한 내용을 악용하거나 최대한 활용하는 데 소비하는 시간의 균형을 맞추는 것입니다.

중앙선을 따르는 AWS DeepRacer 샘플 보상 함수 예제에서 에이전트는 먼저 환경을 탐색하기 위해 무작위 행동을 취합니다. 즉, 트랙의 중심에 잘 머무르지 않습니다. 시간이 흐르면서 에이전트는 어떤 행동이 중앙선 근처에 머무르게 하는지 알아내기 시작하지만, 무작위 동작을 계속 취함으로써 그것을 알아내게 된다면 전체 랩 동안 트랙 중앙 부근에 머무르는 법을 배우는 데 오랜 시간이 걸립니다. 따라서 정책에서 적절한 행동을 학습하기 시작하면 에이전트는 무작위 행동을 취하는 대신 해당 행동을 사용하기 시작합니다. 하지만 에이전트가 항상 좋은 행동을 이용하거나 악용한다면 에이전트는 더 이상 환경을 탐색하지 않기 때문에 새로운 발견을 하지 않을 것입니다. 이러한 절충안을 종종 RL에서의 탐색과 활용 문제라고 합니다.

기본 행동 공간과 샘플 보상 함수를 사용해 실험해 보십시오. 모두 살펴본 후에는 사용자 지정 행동 공간과 사용자 지정 보상 함수를 설계하여 지식을 활용하십시오.

AWS DeepRacer 훈련 알고리즘

프록시멀 정책 최적화(PPO) 대 소프트 액터 크리틱(SAC)

SAC와 PPO 알고리즘은 모두 정책과 가치 함수를 동시에 학습하지만 전략은 크게 세 가지 측면에서 다릅니다.

PPO	SAC
불연속 행동 공간과 연속 행동 공간 모두에서 작동	연속 행동 공간에서 작동
정책 기반	정책 외
엔트로피 정규화 사용	최대화 목표에 엔트로피를 추가합니다.

안정적 vs. 데이터를 많이 사용

환경을 탐색하면서 PPO 및 SAC 알고리즘 정책에 따라 학습한 정보는 다르게 활용됩니다. PPO는 정책에 대한 학습을 사용하는데, 이는 환경에 대한 현행 정책에서 얻은 관찰을 통해 그 가치 함수를 학습한다는 의미입니다. SAC는 정책 외 학습을 사용하므로 이전 정책에서 환경을 탐색하여 얻은 관찰을 활용할 수 있습니다. 정책 외 학습과 정책 기반 학습 간의 균형점은 종종 안정성과 데이터 효율성입니다. 정책 기반 알고리즘은 안정성이 더 높지만 데이터를 많이 사용하는 경향이 있는 반면, 정책 외 알고리즘은 그 반대 경향을 가지고 있습니다.

탐색 및 착취

탐색과 착취는 RL의 주요 과제입니다. 알고리즘은 이전 경험의 알려진 정보를 활용하여 더 높은 누적 보상을 달성해야 하지만, 미래에 최적의 정책을 찾는 데 사용할 수 있는 새로운 경험을 얻기 위한 탐색도 필요합니다. 정책을 여러 번 반복하여 학습하고 환경에 대해 더 많이 알게 되면 주어진 관찰에 맞는 행동을 선택하는 것이 더 확실해집니다. 그러나 정책을 충분히 검토하지 않으면 최적이 아니더라도 이미 학습한 정보를 고수할 가능성이 높습니다. PPO 알고리즘은 에이전트가 로컬 최적값으로 수렴하는 것을 방지하는 엔트로피 정규화를 사용하여 탐색을 장려합니다. SAC 알고리즘은 최대화 목표에 엔트로피를 추가하여 탐색과 활용 사이에서 탁월한 균형을 유지합니다.

Entropy

이러한 맥락에서 “엔트로피”는 정책의 불확실성을 나타내는 척도이므로 정책이 주어진 상태에 맞는 행동을 선택하는 데 얼마나 확신을 갖고 있는지를 나타내는 척도로 해석될 수 있습니다. 엔트로피가 낮은 정책은 어떤 행동을 해야 할지 확신이 있는 반면, 엔트로피가 높은 정책은 어떤 행동을 해야 할지 확신이 서지 않습니다.

SAC 알고리즘의 엔트로피 최대화 전략은 PPO 알고리즘이 엔트로피를 정규화기로 사용하는 것과 비슷한 이점이 있습니다. PPO와 마찬가지로 에이전트가 엔트로피가 더 높은 행동을 선택하도록 장려하여 더 폭넓은 탐색을 장려하고 잘못된 로컬 최적값으로의 수렴을 방지합니다. 엔트로피 조절과 달리 엔트로피 최대화에는 고유한 이점이 있습니다. 위험하지 않은 행동을 선택하는 정책을 포기하는 경향이 있는데, 이는 SAC 알고리즘이 PPO보다 데이터 효율성이 더 높은 경향이 있는 또 다른 이유입니다.

SAC 알파 하이퍼파라미터를 사용하여 SAC의 엔트로피 양을 조정합니다. 최대 SAC 알파 엔트로피 값(1.0)은 탐색에 유리합니다. 최소값(0.0)은 표준 RL 목표를 복원하고 탐색을 장려하는 엔트로피 보너스를 무효화합니다. 실험을 시작하기 좋은 SAC 알파 값은 0.5입니다. 모델을 반복하면서 그에 맞게 조정하십시오.

PPO 알고리즘과 SAC 알고리즘을 모두 사용해 보고, 하이퍼파라미터를 실험하고, 다양한 행동 공간에서 이를 사용해 보십시오.

AWS DeepRacer 솔루션 워크플로

AWS DeepRacer 모델을 훈련할 때는 다음과 같은 일반 작업이 수반됩니다.

1. AWS DeepRacer 서비스가 가상 트랙, 차량의 의미하는 에이전트, 그리고 배경으로 시뮬레이션을 초기화합니다. 에이전트는 정책 신경망을 구체화합니다. 구체화된 정책 신경망은 PPO 알고리즘에서 정의하는 하이퍼파라미터를 사용해 튜닝도 가능합니다.
2. 에이전트가 임의의 상태(전방 카메라에서 촬영되는 이미지)에 따라 (지정된 조향 각도와 속도로) 행동합니다.
3. 시뮬레이션 환경은 에이전트 행동에 따라 에이전트의 위치를 업데이트한 후 보상과 업데이트된 카메라 이미지를 반환합니다. 상태, 행동, 보상 및 새로운 상태의 형식으로 수집되는 경험은 신경망을 주기적으로 업데이트하는 데 사용됩니다. 이렇게 업데이트된 신경망 모델을 사용해 경험이 추가로 생성됩니다.
4. 시뮬레이션 트랙을 따라 에이전트의 1인칭 시점으로 훈련 진행 상황을 모니터링할 수 있습니다. 또한 에피소드당 보상, 손실 함수 값, 정책 엔트로피 같은 지표도 표시할 수 있습니다. 훈련 진행에 따른 CPU 또는 메모리 사용량도 표시됩니다. 그 밖에도 로그가 분석 및 디버깅을 목적으로 자세하게 기록됩니다.

5. AWS DeepRacer 서비스는 신경망 모델을 영구 스토리지에 주기적으로 저장합니다.
6. 훈련은 시간 제한을 기준으로 중지됩니다.
7. 시뮬레이터에서 훈련된 모델을 평가할 수 있습니다. 평가를 원한다면 선택한 트랙에서 선택한 주행 횟수만큼 타임 트라이얼(time trial: 일정한 거리를 개별적으로 주행해 걸린 시간으로 승부를 겨루는 방식)을 할 수 있도록 훈련된 모델을 제출하십시오.

모델 훈련과 평가를 성공적으로 마쳤으면 이제 물리적 에이전트(AWS DeepRacer 차량)에 업로드할 수 있습니다. 업로드 프로세스는 다음과 같은 단계로 구성됩니다.

1. 훈련된 모델을 영구 스토리지(Amazon S3 버킷)에서 다운로드합니다.
2. 차량의 디바이스 제어 콘솔을 사용해 훈련된 모델을 디바이스에 업로드합니다. 콘솔에서 시뮬레이션 행동 공간을 물리적 행동 공간에 매핑할 수 있도록 차량을 보정합니다. 그 밖에 콘솔에서 조절 패리티를 확인하거나, 전방 카메라 피드를 확인하거나, 모델을 추론 엔진에 탑재하거나, 실제 트랙에서 주행하는 차량을 지켜볼 수 있습니다.

   차량의 디바이스 제어 콘솔은 차량의 컴퓨팅 모듈에 호스팅되는 웹 서버입니다. 따라서 Wi-Fi 네트워크가 연결되어 있으면 컴퓨터나 모바일 디바이스의 웹 브라우저를 사용해 차량 IP 주소에서 콘솔에 액세스할 수 있습니다.
3. 다양한 조명과 배터리 레벨, 그리고 표면 질감 및 색상에서 차량을 주행하면서 실험합니다.

   물리적 환경의 디바이스 성능은 모델 제약 또는 부족한 훈련으로 인해 시뮬레이션 환경의 성능과 다를 수 있습니다. 이러한 현상을 sim2real 성능 차이라고 합니다. 성능 차이를 줄이려면 the section called “시뮬레이션 환경과 실제 환경의 성능 차이” 단원을 참조하십시오.

시뮬레이션 환경과 실제 환경의 성능 차이

시뮬레이션에서는 실제 환경의 모든 요소를 정확하게 수집하지 못하기 때문에 시뮬레이션에서 훈련된 모델이 실제 환경에서 성능을 발휘하지 못할 수도 있습니다. 이러한 불일치를 종종 시뮬레이션 환경과 실제 환경의(sim2real) 성능 차이라고 부릅니다.

지금까지 디바이스에서 sim2real 성능 차이를 최소화하기 위한 노력이 많았습니다. 예를 들어 시뮬레이션 에이전트는 초당 약 10회 행동하도록 프로그래밍됩니다. 이는 AWS DeepRacer 디바이스의 추론 실행 횟수인 초당 약 10회의 추론과 일치합니다. 그 밖에도 훈련에서 각 에피소드를 시작할 때 에이전트의 위치는 무작위로 결정됩니다. 그러면 에이전트가 트랙 전체에서 균일하게 학습할 가능성이 극대화되기 때문입니다.

real2sim 성능 차이를 줄이려면 시뮬레이션 트랙과 실제 트랙에서 동일하거나 비슷한 색상, 형상 및 크기를 사용해야 합니다. 시각적 분산을 줄이고 싶다면 실제 트랙 주변에 장벽을 설치하십시오. 또한 훈련에서 사용되는 행동 공간이 실제 환경과 일치할 수 있도록 디바이스의 속도와 조향 각도 범위를 주의하여 보정하십시오. 훈련에서 사용하는 것과 다른 시뮬레이션 트랙에서 모델 성능을 평가하면 real2real 성능 차이가 크게 나타날 수 있습니다.

AWS DeepRacer 모델을 학습할 때 sim2real 격차를 줄이는 방법에 대한 자세한 내용은 the section called “실제 환경에 대한 훈련 최적화”을 참조하십시오.

AWS DeepRacer 시작하기

AWS DeepRacer를 시작하기 위해, 먼저 AWS DeepRacer 콘솔을 사용하여 자율 주행 요구 사항에 적합한 센서로 에이전트를 구성하고, 지정된 센서를 사용하여 에이전트에 대한 강화 학습 모델을 훈련하고, 훈련된 모델을 평가하여 모델 품질을 결정하는 단계를 살펴보겠습니다. 모델을 학습시킨 후에는 모델을 반복하여 레이스에 출전시킬 수 있습니다.

주제

- 첫 번째 AWS DeepRacer 모델 훈련
- 시뮬레이션에서 AWS DeepRacer 모델 평가

첫 번째 AWS DeepRacer 모델 훈련

이 안내에서는 AWS DeepRacer 콘솔을 사용하여 첫 번째 모델을 훈련시키는 방법을 보여줍니다.

AWS DeepRacer 콘솔을 사용하여 강화 훈련 모델을 훈련하려면

모델 훈련 과정을 시작하기 위해 AWS DeepRacer 콘솔에서 모델 생성 버튼을 어디에서 찾을 수 있는지 알아봅니다.

강화 훈련 모델을 훈련하려면

1. AWS DeepRacer를 처음 사용하는 경우, 서비스 랜딩 페이지에서 모델 생성을 선택하거나 기본 탐색 창의 강화 훈련 제목 아래에서 시작하기를 선택합니다.
2. 강화 훈련 시작 페이지의 2단계: 모델 생성 아래에서 모델 생성을 선택합니다.

   또는 기본 탐색 창의 강화 훈련 제목에서 모델을 선택합니다. Your models(모델) 페이지에서 모델 생성을 선택합니다.

모델 이름과 환경을 지정하십시오.

모델 이름을 지정하고 자신에게 적합한 시뮬레이션 트랙을 선택하는 방법을 알아봅니다.

모델 이름 및 환경 지정하기

1. 모델 생성 페이지의 훈련 세부 정보에서 모델 이름을 입력합니다.
2. 훈련 작업 설명을 추가할 수도 있습니다.

3. 선택적 태그 추가에 대해 자세히 알아보려면 태그 지정 단원을 참조하십시오.
4. 환경 시뮬레이션에서 AWS DeepRacer 에이전트의 훈련 환경으로 사용할 트랙을 선택합니다. 트랙 방향에서 시계 방향 또는 시계 반대방향을 선택합니다. 그리고 다음을 선택합니다.

   첫 번째 달리기의 경우 간단한 모양과 부드러운 회전이 가능한 트랙을 선택합니다. 나중에 반복할 때 더욱 복잡한 트랙을 선택하여 모델을 점진적으로 향상시킬 수 있습니다. 특정 레이싱 이벤트에 사용할 목적으로 모델을 훈련하려면 이벤트 트랙과 가장 유사한 트랙을 선택합니다.
5. 페이지 하단에서 다음을 선택합니다.

레이스 유형과 훈련 알고리즘을 선택하십시오.

AWS DeepRacer 콘솔에는 세 가지 레이스 유형과 두 가지 훈련 알고리즘 중에서 선택할 수 있습니다. 기술 수준 및 훈련 목표에 적합한 것이 무엇인지 알아봅니다.

레이스 유형과 트레이닝 알고리즘을 선택하려면

1. 모델 생성 페이지의 레이스 유형에서 타임 트라이얼, 장애물 회피 또는 헤드-투-봇을 선택합니다.

   처음 실행할 때는 타임 트라이얼을 선택하는 것이 좋습니다. 이 레이스 유형에 맞게 에이전트의 센서 구성을 최적화하는 방법에 대한 지침은 the section called “타임 트라이얼을 위한 훈련 조정”을 참조하십시오.
2. 선택 사항으로 이후 실행에서는 장애물 회피를 선택하여 선택한 트랙을 따라 고정 또는 무작위 위치에 배치된 정지 장애물을 우회할 수 있습니다. 자세한 내용은 the section called “장애물 회피 레이스를 위한 훈련 조정” 섹션을 참조하십시오.

   a. 트랙의 두 차선에 걸쳐 사용자가 지정한 고정된 위치에 상자를 생성하려면 고정 위치를 선택하고 훈련 시뮬레이션의 각 에피소드가 시작될 때 두 차선에 걸쳐 무작위로 분포된 장애물을 생성하려면 무작위 위치를 선택합니다.

   b. 다음으로, 트랙에 있는 장애물 수 값을 선택합니다.

   c. 고정 위치를 선택한 경우 트랙에서의 각 장애물 배치를 조정할 수 있습니다. 차선 배치의 경우 안쪽 차선과 바깥쪽 차선 중에서 선택합니다. 기본적으로 장애물은 트랙 전체에 고르게 분포됩니다. 장애물의 시작선과 결승선 사이의 거리를 변경하려면 시작과 끝 사이의 위치(%) 필드에 해당 거리의 7~90 사이의 백분율을 입력합니다.
3. 선택 사항으로 좀 더 야심찬 레이스를 원한다면 헤드-투-봇 레이스를 선택하여 일정한 속도로 움직이는 최대 4대의 로봇 차량과 레이스할 수도 있습니다. 자세한 내용은 the section called “헤드-투-봇 레이스를 위한 훈련 조정”을 참조하십시오.

   a. 로봇 차량 수 선택에서 에이전트가 훈련시킬 로봇 차량 수를 선택합니다.
   b. 다음으로 로봇 차량이 트랙을 돌아다닐 속도를 초당 밀리미터 단위로 선택합니다.
   c. 선택 사항으로 차선 변경 활성화 체크박스를 선택하여 로봇 차량이 1~5초마다 차선을 임의로 변경할 수 있도록 할 수 있습니다.
4. 훈련 알고리즘 및 하이퍼파라미터에서 소프트 액터 크리틱(SAC) 또는 근접 정책 최적화(PPO) 알고리즘을 선택합니다. AWS DeepRacer 콘솔에서는 SAC 모델은 연속 행동 공간에서 훈련해야 합니다. PPO 모델은 연속 또는 이산 행동 공간에서 훈련할 수 있습니다.
5. 모델 생성 페이지의 훈련 알고리즘 및 하이퍼파라미터에서 기본 하이퍼파라미터 값을 그대로 사용합니다.

   나중에 훈련 성과를 향상시키려면 하이퍼파라미터를 확장하고 다음과 같이 기본 하이퍼파라미터 값을 수정합니다.

   a. 경사 하강 배치 크기에서 available options(사용 가능한 옵션)를 선택합니다.
   b. 에포크 수에서 valid value(유효 값)를 설정합니다.
   c. 학습 속도에서 valid value(유효 값)를 설정합니다.
   d. SAC 알파 값(SAC 알고리즘만 해당)의 경우 유효한 값을 설정하십시오.
   e. 엔트로피에서 valid value(유효 값)를 설정합니다.
   f. 할인 계수에서 valid value(유효 값)를 설정합니다.
   g. 손실 유형에서 available options(사용 가능한 옵션)를 선택합니다.
   h. 각 정책 업데이트 반복 간 경험 에피소드 수에서 valid value(유효 값)를 설정합니다.

   하이퍼파라미터에 대한 자세한 내용은 체계적인 하이퍼파라미터 튜닝 단원을 참조하십시오.
6. 다음을 선택합니다.

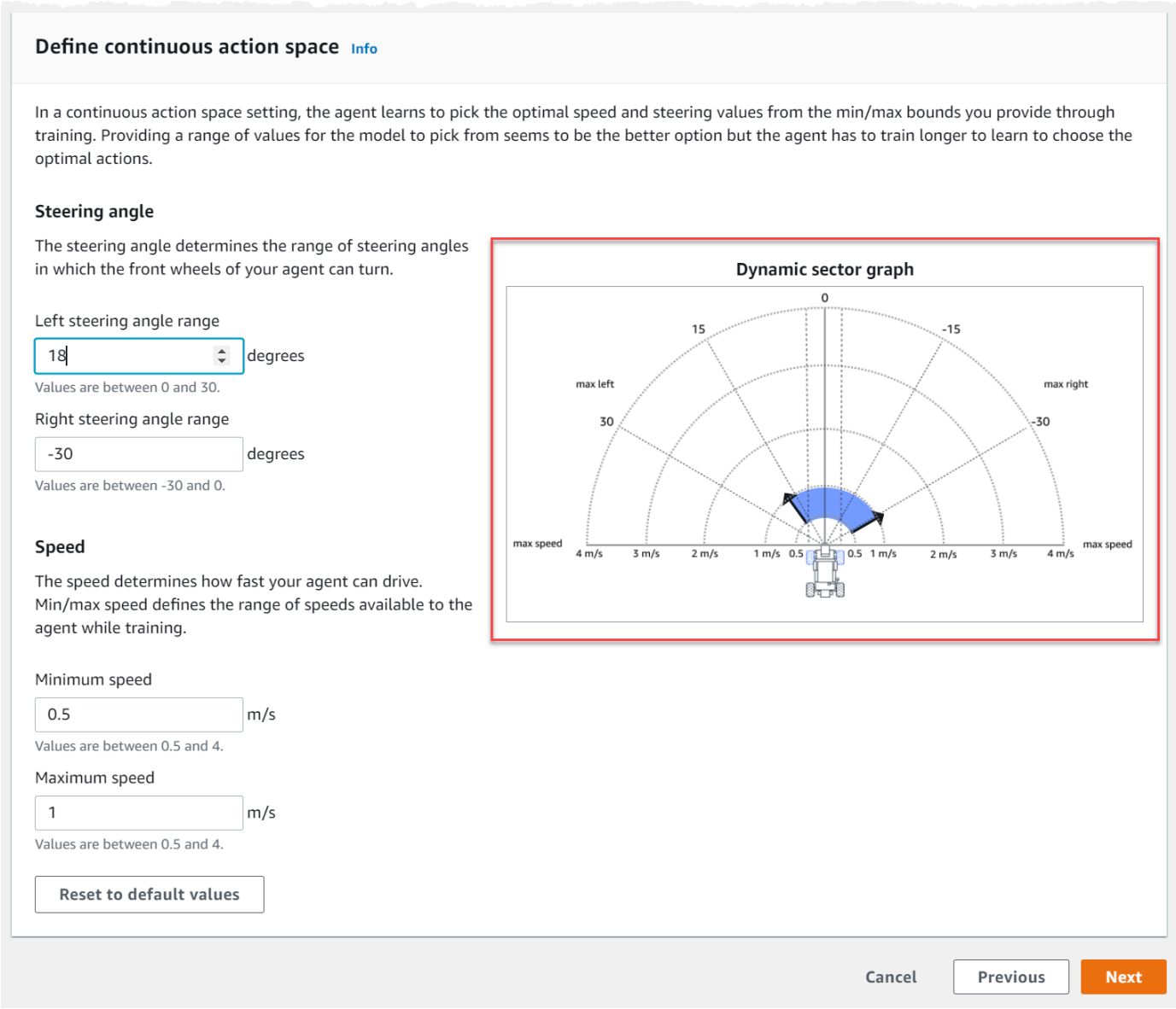
행동 공간 정의

행동 공간 정의 페이지에서 소프트 액터 크리틱(SAC) 알고리즘으로 훈련하도록 선택한 경우 기본 행동 공간은 연속 행동 공간입니다. 근접 정책 최적화(PPO) 알고리즘으로 훈련하기로 선택한 경우 연속 작업 공간과 불연속 작업 공간 중에서 선택합니다. 각 행동 공간 및 알고리즘이 에이전트의 훈련 경험을 어떻게 형성하는지 자세히 알아보려면 the section called “행동 공간 및 보상 함수”을 참조하십시오.

연속 행동 공간 정의하기(SAC 또는 PPO 알고리즘)

1. 연속 행동 공간 정의에서 왼쪽 조향 각도 범위와 오른쪽 조향 각도 범위를 선택합니다.

   각 조향 각도 범위에 대해 다른 각도를 입력하고 범위 변화를 시각화를 관찰하여 동적 섹터 그래프에 선택 사항을 표시해 봅니다.



2. 속도에서 에이전트의 최소 및 최대 속도를 초당 밀리미터 단위로 입력합니다.

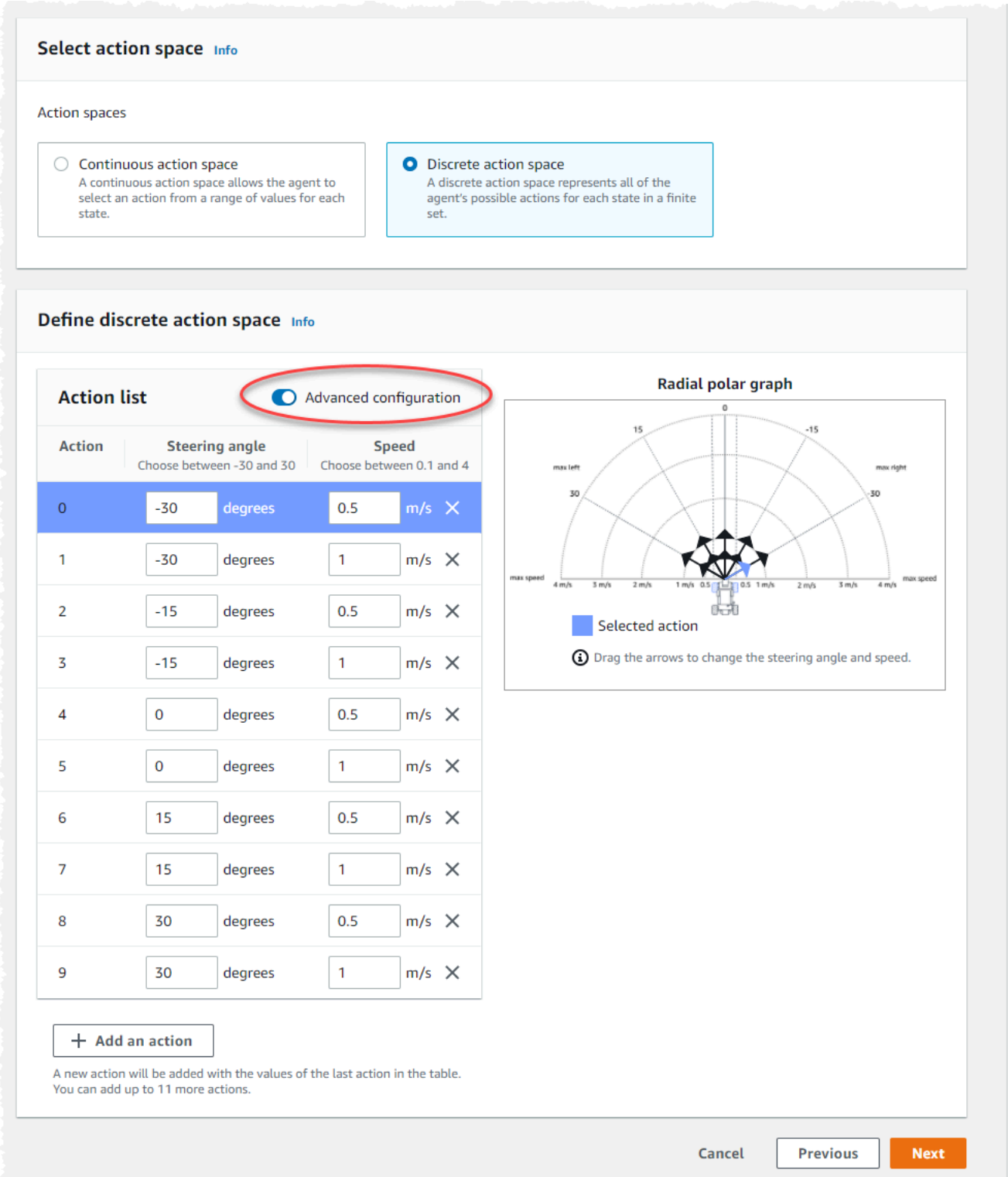
   변경 사항이 동적 섹터 그래프에 어떻게 반영되는지 확인하십시오.

3. 필요에 따라 기본값으로 재설정을 선택하여 원하지 않는 값을 지울 수 있습니다. 그래프에서 다양한 값을 시험해 보면서 실험하고 훈련해 보는 것이 좋습니다.

4. Next(다음)를 선택합니다.

불연속 행동 공간 정의하기(PPO 알고리즘만 해당)

1. 드롭다운 목록에서 조향 각도 세분성 값을 선택합니다.
2. 에이전트의 최대 조향 각도를 1~30 사이의 각도로 선택하십시오.
3. 드롭다운 목록에서 속도 세분성 값을 선택합니다.
4. 에이전트의 최대 속도 값을 0.1~4 사이의 초당 밀리미터 단위로 선택합니다.
5. 작업 목록의 기본 작업 설정을 사용하거나 선택적으로 고급 구성을 켜서 설정을 미세 조정할 수 있습니다. 값을 조정한 후 이전을 선택하거나 고급 구성을 끄면 변경 내용이 손실됩니다.



a. 조향 각도 열에 -30도에서 30도 사이의 값을 입력합니다.

b. 속도 열에 최대 9개의 동작에 대해 초당 0.1~4 밀리미터 사이의 값을 입력합니다.

c. 선택 사항으로 작업 추가를 선택하여 작업 목록의 행 수를 늘릴 수도 있습니다.

d. 필요에 따라 행에서 X를 선택하여 제거할 수도 있습니다.

6. Next(다음)를 선택합니다.

가상 자동차 선택

가상 자동차를 시작하는 방법을 알아봅니다. 매달 Open Division에서 경쟁하여 새로운 커스텀 자동차, 페인트 작업, 개조 작업을 획득하십시오.

가상 자동차를 선택하려면

1. 차량 쉘 및 센서 구성 선택 페이지에서 레이스 유형 및 행동 공간과 호환되는 쉘을 선택하십시오. 차고에 해당 차량이 없는 경우, 기본 탐색 창의 강화 훈련 제목 아래에 있는 내 차고로 이동하여 새로 만드십시오.

   타임 트라이얼 트레이닝의 경우 오리지널 DeepRacer의 기본 센서 구성과 단안 카메라만 있으면 되지만, 다른 모든 쉘 및 센서 구성은 행동 공간이 일치하기만 하면 작동합니다. 자세한 내용은 the section called "타임 트라이얼을 위한 훈련 조정" 섹션을 참조하십시오.

   장애물 회피 훈련에는 스테레오 카메라가 유용하지만 고정된 위치에 고정된 장애물을 피하는 데에 단일 카메라를 사용할 수 있습니다. LiDAR 센서는 선택 사항입니다. the section called "행동 공간 및 보상 함수" 섹션을 참조하십시오.

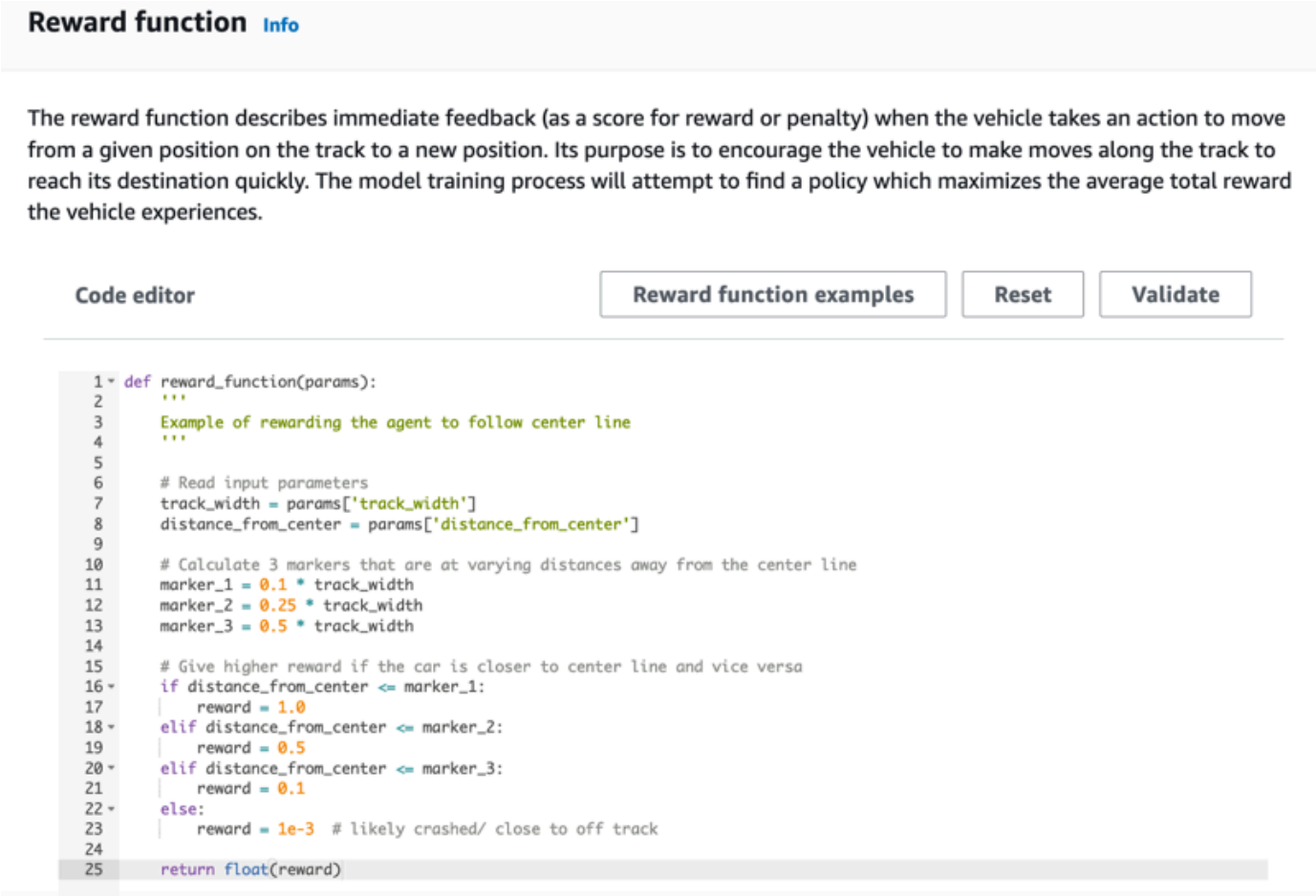
   헤드-투-봇 훈련의 경우 단일 카메라 또는 스테레오 카메라 외에도 LiDAR 장치는 움직이는 다른 차량을 추월하는 동안 사각지대를 감지하고 피하는 데 최적입니다. 자세한 내용은 the section called "헤드-투-봇 레이스를 위한 훈련 조정"을 참조하십시오.
2. Next(다음)를 선택합니다.

보상 함수를 사용자 지정하십시오.

보상 함수는 강화 훈련의 핵심 부분입니다. 차량(에이전트)이 트랙(환경)을 탐색할 때 특정 조치를 취하도록 장려하는 데 이 기능을 사용하는 방법을 알아보십시오. 반려동물의 특정 행동을 장려하거나 억제하는 것과 마찬가지로 이 도구를 사용하여 자동차가 최대한 빨리 랩을 끝내고 트랙을 벗어나거나 물체와 충돌하지 않도록 할 수 있습니다.

보상 함수를 사용자 지정하려면

1. 모델 생성 페이지의 보상 함수에서 첫 번째 모델에 대해 기본 보상 기능 예제를 그대로 사용합니다.



나중에 보상 함수 예제를 선택하여 다른 예제 함수를 선택한 다음 코드 사용를 선택하여 선택한 보상 함수를 수락할 수 있습니다.

시작할 수 있는 함수 예제는 4개입니다. 이들 함수는 트랙 중앙(기본값)을 따르는 방법, 에이전트를 트랙 경계 안에 유지하는 방법, 지그재그 주행을 방지하는 방법, 정지 장애물 또는 기타 이동 차량과 충돌을 방지하는 방법을 보여줍니다.

보상 함수에 대한 자세한 내용은 the section called “보상 함수 참조” 단원을 참조하십시오.

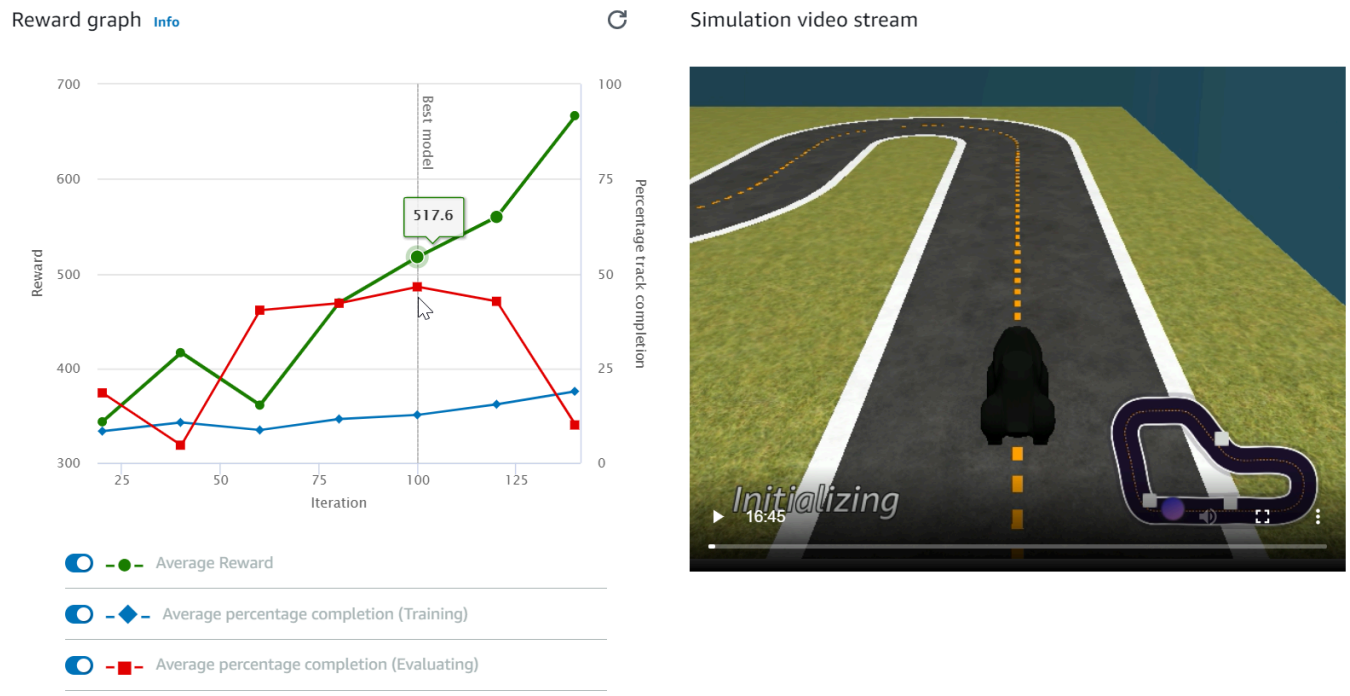
2. 중지 조건 아래에서 기본 최대 시간 값을 그대로 두거나 새 값을 설정해 훈련 작업을 종료하여 장기 실행(및 가능한 런어웨이) 훈련 작업을 방지합니다.

   훈련 초기 단계에서 실험할 때는 이 파라미터를 작은 값으로 시작한 다음 점진적으로 훈련 시간을 늘려야 합니다.

3. AWS DeepRacer에 자동 제출에서는 훈련 완료 후 자동으로 이 모델을 AWS DeepRacer에 제출하고 상품을 받을 기회를 얻기 옵션이 기본적으로 선택되어 있습니다. 선택 사항으로, 체크 표시를 선택하여 모델 입력을 거부할 수도 있습니다.

4. 리그 요구 사항에서 거주 국가를 선택하고 체크박스를 선택하여 이용 약관에 동의하십시오.
5. 모델 생성 페이지에서 모델 생성을 선택하여 모델 생성 및 훈련 작업 인스턴스 프로비저닝을 시작합니다.
6. 제출 후 훈련 작업이 초기화되는 것을 확인한 다음 실행합니다.

   초기화 프로세스에서 초기화 중에서 진행 중으로 변경하는 데 몇 분 걸립니다.
7. 보상 그래프 및 시뮬레이션 비디오 스트림을 시청하여 훈련 작업의 진행 상황을 관찰합니다. 훈련 작업이 완료될 때까지 보상 그래프 옆에 있는 새로 고침 버튼을 주기적으로 선택하여 보상 그래프를 새로 고칠 수 있습니다.



훈련 작업은 AWS 클라우드에서 실행되므로 AWS DeepRacer 콘솔을 열어 둘 필요가 없습니다. 그러나 작업이 진행되는 동안 언제든지 콘솔로 돌아와 모델을 확인할 수 있습니다.

시뮬레이션 비디오 스트림 창이나 보상 그래프 디스플레이가 응답하지 않는 경우 브라우저 페이지를 새로 고쳐 훈련 진행 상황을 업데이트합니다.

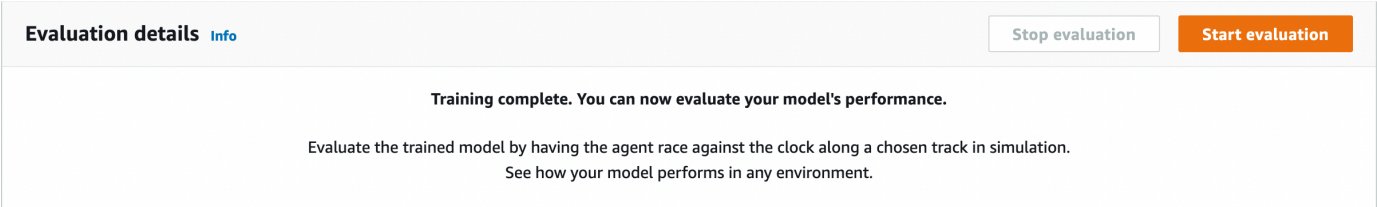
시뮬레이션에서 AWS DeepRacer 모델 평가

훈련 작업이 완료되면 훈련된 모델을 평가하여 수렴 거동을 평가해야 합니다. 평가는 선택한 트랙에서 여러 번 시도를 완료하고 훈련된 모델이 추론할 수 있는 작업에 따라 에이전트가 트랙에서 이동하도록 하여 진행됩니다. 성능 지표에는 트랙 완주율과 각 트랙에서 완주 또는 트랙 이탈까지 주행 시간이 포함됩니다.

훈련된 모델을 평가하기 위해 AWS DeepRacer 콘솔을 사용할 수 있습니다. 이렇게 하려면 이 항목의 단계를 따르십시오.

AWS DeepRacer 콘솔에서 훈련된 모델을 평가하려면

1. https://console.aws.amazon.com/deepracer에서 AWS DeepRacer 콘솔을 엽니다.
2. 기본 탐색 창에서 모델을 선택한 다음 모델 목록에서 방금 훈련한 모델을 선택하여 모델 세부 정보 페이지를 엽니다.
3. 평가 탭을 선택합니다.
4. 평가 세부 사항에서 평가 시작을 선택합니다.

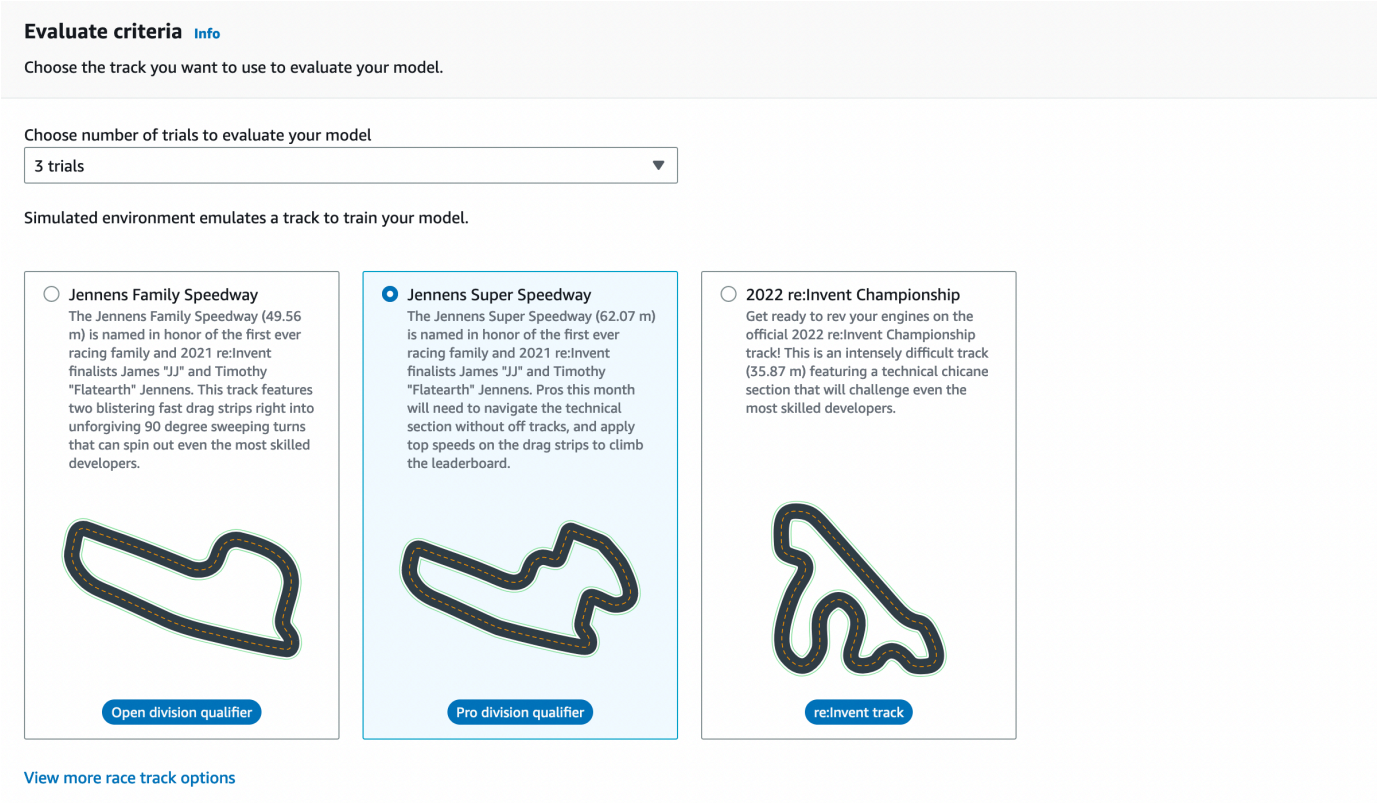


훈련 작업 상태가 완료됨으로 변경되거나, 훈련 작업이 완료되지 않은 경우 모델의 상태가 준비로 변경되면 평가를 시작할 수 있습니다.

모델은 훈련 작업이 완료되었을 때 준비 상태가 됩니다. 훈련이 완료되지 않은 경우 모델이 실패 지점까지 훈련된 경우 준비 상태일 수도 있습니다.

5. 모델 평가 페이지의 레이스 유형에서 평가 이름을 입력하고 모델을 훈련하기 위해 선택한 레이스 유형을 선택합니다.

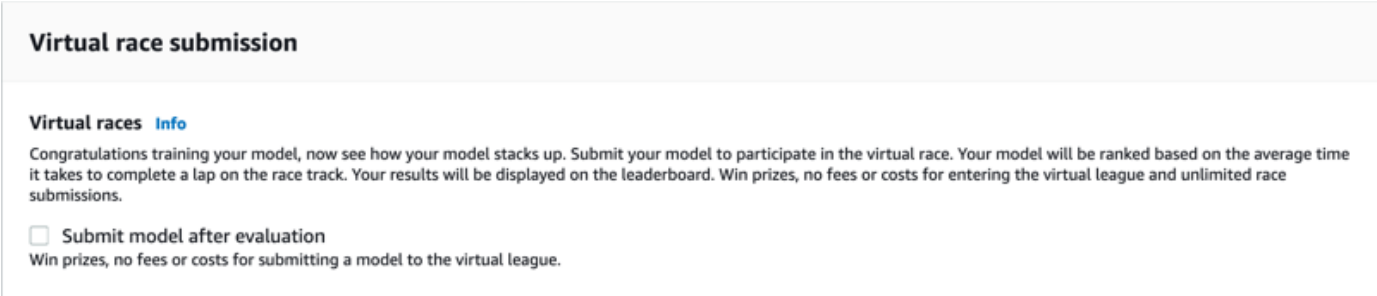
   평가를 위해 훈련에 사용한 레이스 유형과 다른 레이스 유형을 선택할 수 있습니다. 예를 들어 헤드-투-봇 레이스에 대해 모델을 훈련한 다음 타임 트라이얼에 대해 모델을 평가할 수 있습니다. 일반적으로 훈련 레이스 유형이 평가 레이스 유형과 다른 경우 모델이 잘 일반화해야 합니다. 첫 실행의 경우 평가와 훈련 모두에 동일한 레이스 유형을 사용해야 합니다.
6. 모델 평가 페이지의 평가 기준에서 실행하려는 시도 횟수를 선택하고 모델을 평가할 트랙을 선택합니다.



일반적으로 모델 훈련에 사용한 것과 동일하거나 비슷한 트랙을 선택합니다. 모델 평가를 위해 아무 트랙이나 선택할 수 있지만 훈련에 사용한 것과 가장 비슷한 트랙을 선택해야만 최적의 성능을 기대할 수 있습니다.

모형이 잘 일반화되는지 확인하려면 훈련에 사용된 것과 다른 평가 트랙을 선택합니다.

7. 모델 평가 페이지의 가상 레이스 제출에서 첫 번째 모델에 대해 평가 후 모델 제출 옵션을 끕니다. 나중에 레이싱 이벤트에 참가하려면 이 옵션을 활성화한 상태로 둡니다.



8. 모델 평가 페이지에서 평가 시작을 선택하여 평가 작업 생성 및 초기화를 시작합니다.

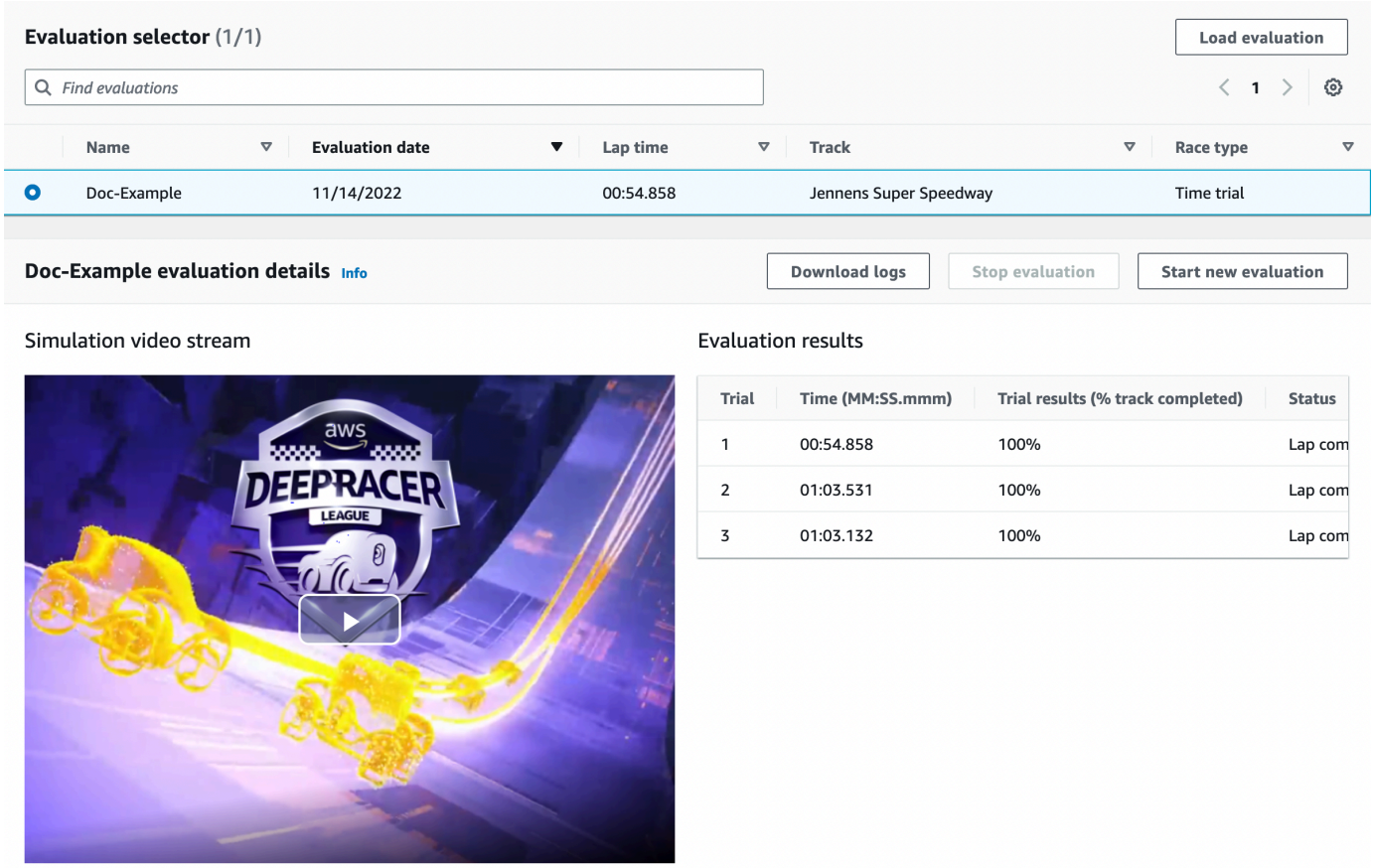
초기화 프로세스는 완료할 때까지 약 3분 걸립니다.

9. 평가가 진행됨에 따라 시도 시간 및 트랙 완주율을 포함한 평가 결과가 평가 아래에 표시됩니다. 시뮬레이션 비디오 스트림 창에서 에이전트가 선택한 트랙에서 어떤 성능을 보이는지 확인할 수 있습니다.

   평가 작업이 완료되기 전에 중지할 수 있습니다. 평가 작업을 중지하려면 평가 카드 오른쪽 상단의 평가 중지를 선택한 다음 확인을 통해 평가를 중지하십시오.

10. 평가 작업이 완료된 후 평가 결과에서 모든 시도의 성능 지표를 검사합니다. 함께 제공되는 시뮬레이션 비디오 스트림은 더 이상 사용할 수 없습니다.

    평가 선택기에서 모델 평가 기록을 확인할 수 있습니다. 특정 평가의 세부 정보를 보려면 평가 선택기 목록에서 평가를 선택한 다음 평가 선택기 카드의 오른쪽 상단에서 평가 로딩을 선택합니다.



이 평가 작업의 경우 훈련된 모델은 상당한 시간 지연 페널티를 감수하고 시도를 완료합니다. 첫 번째 실행에서 이것은 드문 일이 아닙니다. 가능한 이유는 훈련이 수렴하지 않았고 훈련에 더 많은 시간이 필요하거나, 에이전트가 반응할 수 있는 여지를 늘려야 하거나, 다양한 환경을 처리하도록 보상 함수를 업데이트해야 하는 것이 포함됩니다.

이전에 훈련한 모델을 복제하고, 보상 함수를 수정하고, 하이퍼파라미터를 조정한 다음 전체 보상이 수렴되어 성능 지표가 개선될 때까지 프로세스를 반복하면 모델을 개선할 수 있습니다. 훈련 개선 방법에 대한 자세한 내용은 모델 훈련 및 평가 단원을 참조하십시오.

실제 환경에서 주행할 목적으로 완전하게 훈련된 모델을 AWS DeepRacer 디바이스로 이전하려면 모델 아티팩트를 다운로드해야 합니다. 다운로드하려면 모델의 세부 정보 페이지에서 모델 다운로드를 선택합니다. 실제 AWS DeepRacer 디바이스는 새 센서를 지원하지 않고 모델이 새로운 센서 유형에 대해 훈련을 받은 경우 실제 환경에서 AWS DeepRacer 디바이스에 해당 모델을 사용할 때 오류 메시지가 표시됩니다. 물리적 에이전트를 사용하여 AWS DeepRacer 모델을 테스트하는 방법에 대한 자세한 내용은 차량 작동 단원을 참조하십시오.

AWS DeepRacer 리그 레이싱 이벤트 또는 AWS DeepRacer 커뮤니티 레이스에서 지정한 트랙과 동일하거나 비슷한 트랙에서 모델을 훈련한 경우 AWS DeepRacer 콘솔의 가상 레이스에 모델을 제출할 수 있습니다. 이를 위해서는 기본 탐색 창에서 AWS 가상 서킷 또는 커뮤니티 레이스를 따르십시오. 자세한 내용은 레이스 참가 섹션을 참조하십시오.

장애물 회피 또는 헤드-투-봇 레이스를 위한 모델을 훈련하려면 모델과 실제 차량에 새 센서를 추가해야 할 수 있습니다. 자세한 내용은 the section called “레이스 유형 및 이를 가능하게 하는 센서 이해” 섹션을 참조하십시오.

AWS DeepRacer 모델 훈련 및 평가

AWS DeepRacer 차량이 트랙을 따라 스스로 주행할 때는 전면에 장착된 카메라를 사용해 환경 상태를 수집하여 관측 결과에 따라 행동합니다. AWS DeepRacer 모델은 관측 결과와 행동을 예상되는 보상으로 매핑하는 역할을 합니다. 모델 훈련이란 예상되는 보상을 최대화하는 함수를 찾거나 학습하는 과정을 말합니다. 이러한 과정을 통해 최적화된 모델은 차량이 출발선부터 결승선까지 트랙을 따라 스스로 주행하는 데 필요한 행동(속도와 조향 각도 쌍)을 결정합니다.

실제로 함수는 신경망을 통해 표현되며, 이러한 신경망 훈련은 관측된 환경 상태의 시퀀스와 여기에 반응하는 차량의 행동을 고려하여 최적의 신경망 가중치를 찾는 작업이 수반됩니다. 기본적인 최적성 기준은 차량이 교통 사고나 법규 위반 없이 합법적이고 생산적인 주행을 할 수 있도록 유도하는 보상 함수로 설명됩니다. 간단한 보상 함수는 차량이 트랙을 주행할 경우 보상 점수 0을, 트랙을 벗어날 경우 보상 점수 -1을, 그리고 결승선에 도달하면 보상 점수 +1을 반환합니다. 이 보상 함수를 사용하면 트랙에서 벗어날 경우 패널티를 받고 목표 지점에 도달하면 보상을 받습니다. 시간이나 속도가 문제가 되지 않는 경우 이 함수는 좋은 보상이 될 수 있습니다.

직선 트랙에서 벗어나지 않고 최대한 빨리 차량을 운전해야 하는 경우를 가정해 보십시오. 차량 속도가 증가와 감소를 반복하면서 차량은 장애물을 피하거나 트랙에서 벗어나지 않기 위해 왼쪽이나 오른쪽으로 조향하기도 합니다. 고속으로 너무 크게 회전하면 차량이 트랙에서 쉽게 이탈할 수 있습니다. 또한 너무 작게 회전하면 장애물이나 다른 차량과 충돌하는 것을 피하지 못할 수도 있습니다. 일반적으로 비교적 낮은 속도로 크게 회전하거나, 혹은 급커브에서 조향 각도를 줄여야만 최적의 행동이라고 할 수 있습니다. 이러한 동작을 유도하려면 보상 함수가 높은 속도에서 작은 회전을 할 경우 양의 점수를 할당하여 보상하고, 높은 속도에서 큰 회전을 할 경우 음의 점수를 할당하여 벌점을 부과해야 합니다. 마찬가지로 직선 구간에서 속도를 높이거나, 혹은 주변에 장애물이 있을 경우 속도를 낮추면 양의 보상 점수를 반환할 수 있습니다.

보상 함수는 AWS DeepRacer 모델에서 중요한 역할을 합니다. AWS DeepRacer 모델을 훈련할 때는 보상 함수가 반드시 필요합니다. 훈련할 때는 시작부터 끝까지 트랙을 따라 에피소드가 반복됩니다. 에피소드에서는 에이전트가 트랙과 상호 작용하면서 예상되는 누적 보상을 최대화하여 최적의 행동을 학습합니다. 결국, 훈련은 강화 학습 모델을 산출합니다. 훈련이 끝나면 에이전트가 모델에 대해 추론을 실행하여 임의 상태일 때 최적의 행동을 보이면서 자율 주행을 실행합니다. 이러한 훈련은 시뮬레이션 환경에서 가상 에이전트를 사용하거나, 혹은 실제 환경에서 AWS DeepRacer 모형 차량 같은 물리적 에이전트를 사용해 가능합니다.

실제로 강화 학습 모델을 훈련하려면 학습 알고리즘을 선택해야 합니다. 현재 AWS DeepRacer 콘솔은 근위 정책 최적화(PPO) 및 소프트 액터 크리틱(SAC) 알고리즘만 지원합니다. 그런 다음 처음부터 새로 작성하려는 경우가 아니라면 선택한 알고리즘을 지원하는 딥 러닝 프레임워크를 선택할 수 있습니

다. AWS DeepRacer는 SageMaker AI와 통합되어 TensorFlow와 같은 인기 있는 딥 러닝 프레임워크를 AWS DeepRacer 콘솔에서 즉시 사용할 수 있도록 합니다. 프레임워크를 사용하면 훈련 작업 구성 및 실행을 간소화하고 문제와 관련된 보상 함수를 작성 및 개선하는 데 집중할 수 있습니다.

강화 학습 모델의 훈련은 반복 프로세스입니다. 먼저, 한 환경에서 에이전트의 모든 중요한 동작을 한 번에 포함할 수 있는 보상 함수를 정의하기는 어렵습니다. 두 번째로, 하이퍼파라미터는 일반적으로 만족스러운 훈련 성능을 보장하기 위해 튜닝됩니다. 둘 다 실험이 필요합니다 신중한 접근 방식은 간단한 보상 함수로 시작한 다음 점진적으로 개선하는 것입니다. AWS DeepRacer는 훈련된 모델을 복제한 후 이를 사용하여 다음 교육 단계를 바로 시작할 수 있도록 함으로써 이러한 반복 프로세스를 용이하게 합니다. 반복할 때마다 보상 함수에 하나 이상의 정교한 처리를 도입하여 이전에 무시된 변수를 처리하거나 결과가 수렴될 때까지 하이퍼파라미터를 체계적으로 조정할 수 있습니다.

기계 학습에서 일반적인 방법처럼 훈련된 강화 학습 모델을 평가하여 실제 상황에서 추론을 실행할 수 있도록 물리적 에이전트에 배포하기 전에 효과를 확인해야 합니다. 자율 주행의 경우 평가는 시작부터 끝까지 차량이 지정된 트랙을 주행한 빈도 또는 트랙을 벗어나지 않고 코스를 완주한 시간을 기반으로 할 수 있습니다. AWS DeepRacer 시뮬레이션에서는 성능 평가 후 다른 AWS DeepRacer 사용자가 훈련한 모델과 비교할 목적으로 성능 지표를 리더보드에 게시하는 것도 가능합니다.

주제

- AWS DeepRacer가 지원하는 레이스 유형 및 이를 가능하게 하는 센서 이해
- AWS DeepRacer 콘솔을 사용하여 AWS DeepRacer 모델을 훈련 및 평가하십시오.
- AWS DeepRacer 보상 함수 참조

AWS DeepRacer가 지원하는 레이스 유형 및 이를 가능하게 하는 센서 이해

AWS DeepRacer에서는 다음과 같은 유형의 레이싱 이벤트에 참여할 수 있습니다.

- 타임 트라이얼: 방해 받지 않는 트랙에서 시간을 다투어 경주하며 가장 빠른 랩 타임을 목표로 합니다.
- 장애물 회피: 정지 장애물이 있는 트랙에서 시간을 다투어 경주하며 가장 빠른 랩 타임을 목표로 합니다.
- 헤드-투-봇 레이스: 같은 트랙에서 하나 이상의 다른 자동차를 상대로 경주하고 다른 자동차보다 먼저 결승선을 통과하는 것을 목표로 합니다.

AWS DeepRacer 커뮤니티 레이스는 현재 타임 트라이얼만 지원합니다.

AWS DeepRacer 차량에 다양한 센서를 장착하고 실험해야 특정 레이스 유형에서 주변 환경을 관찰할 수 있는 충분한 기능을 제공할 수 있습니다. 다음 단원에서는 지원되는 유형의 자율 레이싱 이벤트를 가능하게 하는 AWS DeepRacer 지원 센서에 대해 설명합니다.

주제

- AWS DeepRacer 레이싱 유형에 맞는 센서 선택
- AWS DeepRacer 모델 교육을 위한 에이전트 구성
- 타임 트라이얼을 위한 AWS DeepRacer 훈련 조정
- 장애물 회피 레이스를 위한 AWS DeepRacer 훈련 조정
- 헤드-투-봇 레이스를 위한 AWS DeepRacer 훈련 조정

AWS DeepRacer 레이싱 유형에 맞는 센서 선택

AWS DeepRacer 차량에는 전방 단안 카메라가 기본 센서로 제공됩니다. 또 하나의 전방 단안 카메라를 추가하여 전방 스테레오 카메라를 만들거나 단안 카메라 또는 스테레오 카메라를 LiDAR 장치로 보완할 수 있습니다.

다음 목록에는 AWS DeepRacer 지원 센서의 기능이 간단한 비용 혜택 분석과 함께 요약되어 있습니다.

전방 카메라

전방 단안 카메라는 트랙 경계 및 형태를 포함하여 호스트 차량 앞의 환경 이미지를 캡처할 수 있습니다. 가장 저렴한 센서이며 잘 표시된 트랙에서 장애물 없는 타임 트라이얼과 같은 간단한 자율 주행 작업을 처리하는 데 적합합니다. 적절한 훈련을 통해 트랙에 고정된 정지 장애물을 회피할 수 있습니다. 그러나 장애물 위치 정보는 훈련된 모형에 내장되어 있으므로 모형이 과적합될 가능성이 있으며 다른 장애물 배치로 일반화되지 않을 수 있습니다. 정지 객체가 무작위 위치에 배치되거나 트랙의 다른 이동 차량에 배치되면 모델이 수렴하지 않을 수 있습니다.

실제로 AWS DeepRacer 차량은 기본 센서로 전방 단안 카메라와 함께 제공됩니다. 이 카메라에는 120도 광각 렌즈가 장착되어 있으며 RGB 이미지를 캡처한 다음 15fps(초당 프레임)로 160 x 120픽셀 회색조 이미지로 변환합니다. 이러한 센서 특성은 시뮬레이터에 보존되어 훈련된 모델이 시뮬레이션에서 실제 세계로 잘 이전 될 가능성을 극대화합니다.

전방 스테레오 카메라

스테레오 카메라에는 동일한 해상도 및 주파수로 이미지를 캡처하는 두 개 이상의 렌즈가 있습니다. 두 렌즈의 이미지는 관찰된 객체의 깊이를 측정하는 데 사용됩니다. 스테레오 카메라의 깊이 정

보는 호스트 차량이 특히 역동적인 환경에서 장애물 또는 다른 전방 차량과의 충돌을 회피하는 데 유용합니다. 그러나 추가된 깊이 정보로 인해 훈련이 더 느리게 수렴됩니다.

실제 AWS DeepRacer 차량에서는 또 하나의 단안 카메라를 추가하여 차량 좌우 양측에 카메라를 장착하여 중안 스테레오 카메라를 구성합니다. AWS DeepRacer 소프트웨어는 두 카메라에서 캡처한 이미지를 동기화합니다. 캡처된 이미지는 회색조로 변환되고, 누적되고, 추론을 위해 신경망으로 공급됩니다. 실제 환경에 잘 일반화하도록 모델을 훈련하기 위해 동일한 메커니즘이 시뮬레이터에 복제됩니다.

LiDAR 센서

LiDAR 센서는 회전 레이저를 사용하여 가시 스펙트럼을 초과하는 광 펄스를 발출하고 복귀하는 데 걸리는 시간을 보냅니다. 특정 펄스 히트가 있는 객체의 방향 및 거리는 LiDAR 장치를 중심으로 한 대형 3D 맵에 점으로 기록됩니다.

예를 들어 LiDAR는 차량이 차선을 변경하는 동안 충돌을 회피하기 위해 호스트 차량의 사각 지대를 탐지하는 데 도움을 줍니다. LiDAR와 모노 또는 스테레오 카메라를 결합하면 호스트 차량이 적절한 조치를 취하기에 충분한 정보를 캡처할 수 있습니다. 그러나 LiDAR 센서는 카메라에 비해 비용이 많이 듭니다. 또한 신경망이 LiDAR 데이터를 해석하는 방법을 배워야 합니다. 따라서 훈련이 수렴하는 데 더 오래 걸립니다.

실제 AWS DeepRacer 차량에서는 LiDAR 센서가 후면에 장착되고 6도 아래로 기울어져 있습니다. 이 센서는 초당 10회 회전의 각속도로 회전하며 가시 범위는 15cm~2m입니다. 호스트 차량의 뒤쪽과 옆에 있는 객체뿐 아니라 앞쪽에서 차량에 가리지 않는 높은 객체도 감지할 수 있습니다. 각도 및 범위는 LiDAR 장치가 환경 잡음에 덜 취약하도록 선택됩니다.

다음과 같은 지원되는 센서 조합으로 AWS DeepRacer 차량을 구성할 수 있습니다.

- 전방 단안 카메라 단독.

  이 구성은 타임 트라이얼뿐 아니라 고정 위치 장애물 회피에 적합합니다.
- 전방 스테레오 카메라 단독.

  이 구성은 장애물이 고정 또는 무작위 위치에 있는 장애물 회피에 적합합니다.
- 전방 단안 카메라와 LiDAR.

  이 구성은 장애물 회피 또는 헤드-투-봇 레이스에 적합합니다.
- 전방 스테레오 카메라와 LiDAR.

이 구성은 장애물 회피 또는 헤드-투-봇 레이스에 적합하지만 타임 트라이얼에는 가장 경제적인 것은 아닙니다.

AWS DeepRacer 차량을 타임 트라이얼에서 장애물 회피, 헤드-투-봇 레이스로 전환하기 위해 더 많은 센서를 추가함에 따라 차량은 훈련 시 기본 신경망에 공급하기 위해 더 많은 환경 데이터를 수집합니다. 그러면 모델이 증가한 복잡성을 처리해야 하므로 훈련이 더욱 어려워집니다. 결국 모델 훈련 방법을 배우는 작업이 더욱 까다로워집니다.

점진적으로 배우려면 먼저 타임 트라이얼 훈련을 시작한 다음 장애물 회피, 그런 다음 헤드-투-봇 레이스로 전환해야 합니다. 자세한 권장 사항은 다음 단원에서 확인할 수 있습니다.

AWS DeepRacer 모델 교육을 위한 에이전트 구성

AWS DeepRacer 차량이 장애물 회피 또는 헤드-투-봇 레이스에서 경주할 수 있도록 강화 학습 모델을 훈련하려면 적절한 센서를 사용하여 에이전트를 구성해야 합니다. 간단한 타임 트라이얼의 경우 단안 카메라로 구성된 기본 에이전트를 사용할 수 있습니다. 에이전트를 구성할 때 선택한 센서와 더 잘 연동하여 의도한 운전 요구 사항을 충족하도록 행동 공간을 사용자 정의하고 신경망 토폴로지를 선택할 수 있습니다. 또한 훈련 중에 시각적 식별을 위해 에이전트의 모양을 변경할 수 있습니다.

완료된 에이전트 구성은 훈련 및 평가를 위해 모델 메타데이터의 일부로 기록됩니다. 평가를 위해 에이전트는 지정된 센서, 행동 공간 및 신경망 기술을 사용하기 위해 기록된 구성을 자동으로 검색합니다.

이 단원에서는 AWS DeepRacer 콘솔에서 에이전트를 구성하는 단계를 안내합니다.

AWS DeepRacer 콘솔에서 AWS DeepRacer 에이전트를 구성하려면

1. AWS DeepRacer 콘솔에 로그인합니다.
2. 기본 탐색 창에서 차고를 선택합니다.
3. 차고를 처음 사용하는 경우 차고에 오신 것을 환영합니다 대화 상자가 표시됩니다. > 또는 <를 선택하여 AWS DeepRacer 차량에 지원되는 다양한 센서에 대한 소개를 살펴보거나 X를 선택하여 대화 상자를 닫습니다. 이 소개 정보는 차고 도움말 창에서 찾을 수 있습니다.
4. 차고 페이지에서 새 차량 빌드를 선택합니다.
5. 차량 수정 페이지의 사양 수정 아래에서 하나 이상의 센서를 선택하여 원하는 레이스 유형에 맞는 최상의 조합을 훈련해 봅니다.

   AWS DeepRacer 차량 타임 트라이얼을 훈련하려면 카메라를 선택합니다. 장애물 회피 또는 헤드-투-봇 레이스의 경우 다른 센서 유형을 사용하는 것이 좋습니다. 스테레오 카메라를 선택하려면 단

안 카메라를 추가로 구입해야 합니다. AWS DeepRacer는 스테레오 카메라를 두 대의 단안 카메라로 만듭니다. 한 대의 차량에 하나의 단안 카메라 또는 하나의 중안 스테레오 카메라를 구성할 수 있습니다. 두 경우 모두, 훈련된 모델이 장애물 회피 또는 헤드-투-봇 레이스의 사각 지대를 감지하고 회피할 수 있도록 하려면 에이전트에 LiDAR 센서를 추가할 수 있습니다.

6. 차고 페이지의 신경망 토폴로지 아래에서 지원되는 네트워크 토폴로지를 선택합니다.

   일반적으로 더 깊은 신경망(계층이 더 많음)은 급한 커브와 방향 회전이 많은 보다 복잡한 트랙을 주행하거나, 정지 장애물을 회피하는 레이스를 하거나, 다른 이동 차량과 경쟁하는 데 더 적합합니다. 그러나 더 깊은 신경망은 훈련 비용이 많이 들고 모델이 수렴하는 데 더 오래 걸립니다. 반면에 얕은 네트워크(계층이 더 적음)는 비용이 적게 들고 훈련하는 데 걸리는 시간이 더 짧습니다. 훈련된 모델은 장애물이 없는 트랙에서 경쟁자 없는 타임 트라이얼과 같은 보다 단순한 트랙 조건이나 주행 요구 사항을 처리할 수 있습니다.

   특히 AWS DeepRacer는 3계층 CNN 또는 5계층 CNN을 지원합니다.

7. 차고 페이지에서 다음을 선택하여 에이전트의 행동 공간 설정으로 이동합니다.
8. 행동 공간 페이지에서 첫 번째 훈련의 기본 설정을 그대로 둡니다. 후속 훈련에서는 조향 각도, 최고 속도 및 세분화에 대해 다양한 설정을 실험합니다. 그리고 다음을 선택합니다.
9. 눈에 띄는 차량 색상 지정 페이지에서 DeepRacer 이름 지정에 이름을 입력한 다음 차량 색상 목록에서 에이전트의 색상을 선택합니다. 그런 다음 제출을 선택합니다.
10. 차고 페이지에서 새 에이전트의 설정을 검토합니다. 추가로 수정하려면 차량 수정을 선택하고 이전 단계를4단계부터 반복합니다.

이제 에이전트를 훈련할 준비가 되었습니다.

타임 트라이얼을 위한 AWS DeepRacer 훈련 조정

AWS DeepRacer를 처음 사용하는 경우 간단한 타임 트라이얼부터 시작하여 차량을 운전할 AWS DeepRacer 모델을 훈련하는 방법을 익힐 수 있습니다. 이렇게 하면 보상 함수, 에이전트, 환경 등의 기본 개념을 차근차근 접할 수 있습니다. 목표는 차량이 트랙을 벗어나지 않고 가능한 한 빨리 랩을 끝낼 수 있는 모델을 훈련하는 것입니다. 그런 다음 훈련된 모델을 AWS DeepRacer차량에 배치하여 추가 센서 없이 실제 트랙에서 주행을 테스트할 수 있습니다.

이 시나리오에 맞게 모델을 학습시키려면 AWS DeepRacer 콘솔의 차고에서 기본 에이전트를 선택할 수 있습니다. 기본 에이전트는 전방 단안 카메라, 기본 행동 공간 및 기본 신경망 토폴로지로 구성되었습니다. 좀 더 정교한 에이전트로 전환하기 전에 기본 에이전트로 AWS DeepRacer 모델 훈련을 시작하는 것이 좋습니다.

기본 에이전트를 사용하여 모델을 훈련하려면 다음 권장 사항을 따르십시오.

1. 보다 규칙적인 형태와 보다 완만한 커브 간단한 트랙으로 모델 훈련을 시작합니다. 기본 보상 함수를 사용합니다. 그리고 30분 동안 모델을 훈련합니다. 훈련 작업이 완료된 후 동일한 트랙에서 모델을 평가하여 에이전트가 랩을 마칠 수 있는지 확인합니다.
2. 보상 함수 매개 변수를 참조하십시오. 다른 인센티브로 훈련을 계속하여 에이전트가 더 빨리 진행하도록 보상합니다. 다음 모델의 훈련 시간을 1~2시간으로 연장합니다. 첫 번째 훈련과 두 번째 훈련 사이의 보상 그래프를 비교합니다. 보상 그래프가 개선되지 않을 때까지 실험을 계속합니다.
3. 행동 공간에 대해 자세히 알아보십시오. 최고 속도(예를 들어, 1m/s)를 높여 모델을 3번째로 훈련합니다. 행동 공간을 수정하려면 수정할 기회가 생겼을 때 차고에서 새 에이전트를 빌드해야 합니다. 에이전트의 최고 속도를 업데이트할 때 알아야 할 점은 최고 속도가 높을수록 에이전트가 평가 중인 트랙을 더 빨리 완료할 수 있고 AWS DeepRacer 차량이 물리적 트랙에서 랩을 더 빨리 마칠 수 있다는 것입니다. 그러나 최고 속도가 높을수록 훈련이 수렴하는 데 걸리는 시간이 길어지는 경우가 많습니다. 에이전트가 커브에서 오버슈트하여 트랙을 벗어날 가능성이 높기 때문입니다. 세분성을 줄여 에이전트가 가속 또는 감속할 여지를 늘리고 다른 방법으로 보상 함수를 조정하여 훈련이 더 빨리 수렴되도록 할 수 있습니다. 훈련이 수렴된 후 3번째 모델을 평가하여 랩 타임이 향상되는지 확인합니다. 더 이상 개선이 없을 때까지 계속 탐색하십시오.
4. 더 복잡한 트랙을 선택하여 1단계부터 3단계까지 반복합니다. 모델을 훈련하는 데 사용한 트랙과 다른 트랙에서 모델을 평가하여 모델이 다른 가상 트랙으로 얼마나 잘 일반화하는지 확인합니다실제 환경으로 일반화.
5. (선택 사항) 하이퍼파라미터 값을 여러 개 사용하여 훈련 프로세스를 개선하고 1단계부터 3단계까지 반복합니다.
6. (선택 사항) AWS DeepRacer 로그를 확인하고 분석합니다. 로그를 분석하는 데 사용할 수 있는 샘플 코드는 https://github.com/aws-samples/aws-deepracer-workshops/tree/master/log-analysis를 참조하십시오.

장애물 회피 레이스를 위한 AWS DeepRacer 훈련 조정

타임 트라이얼에 익숙해지고 몇 가지 수렴된 모델을 훈련한 후에는 보다 어려운 다음 도전인 장애물 회피로 전환합니다. 여기서 목표는 트랙에 배치된 객체와 충돌하지 않고 최대한 빨리 랩을 완주할 수 있는 모델을 훈련하는 것입니다. 이것은 분명히 에이전트가 학습하기 어려운 문제이며 훈련이 수렴하는 데 더 오래 걸립니다.

AWS DeepRacer 콘솔은 두 유형의 장애물 회피 훈련을 지원합니다. 즉, 장애물을 트랙을 따라 고정 또는 무작위 위치에 배치할 수 있습니다. 고정 위치에서는 장애물이 훈련 작업 전반에 걸쳐 동일한 장소에 고정되어 있습니다. 무작위 위치에서는 장애물이 에피소드마다 임의로 위치를 변경합니다.

시스템 자유도가 낮기 때문에 고정 위치 장애물 회피를 위한 훈련이 수렴하기 더 쉽습니다. 그러나 훈련된 모형에 위치 정보가 내장된 경우 모형이 과적합될 수 있습니다. 결과적으로 모형이 과적합되어 잘 일반화되지 않을 수 있습니다. 무작위 위치 장애물 회피의 경우, 에이전트는 이전에 본 적이 없는 위치에서 장애물과 충돌하지 않도록 학습해야 하기 때문에 훈련이 수렴하기가 더 어렵습니다. 그러나 이 옵션을 사용하여 훈련된 모델은 더 잘 일반화되고 실제 레이스에 잘 이전되는 경향이 있습니다. 시작하려면 장애물을 고정 위치에 배치하고 동작에 익숙해진 다음 무작위 위치에 도전하십시오.

AWS DeepRacer 시뮬레이터에서 장애물은 AWS DeepRacer 차량의 포장 상자와 동일한 치수(9.5인치(L) x 15.25인치(W) x 10/5인치(H))의 입방체 상자입니다. 따라서 포장 상자를 실제 트랙에 장애물로 배치하면 시뮬레이터에서 훈련된 모델이 실제 세계로 쉽게 이전될 수 있습니다.

장애물 회피를 실험하려면 아래 단계에 설명된 권장 방법을 따르십시오.

1. 기존 에이전트를 사용자 정의하거나 새 에이전트를 빌드하여 새 센서 및 행동 공간에서 기본 에이전트 또는 실험을 사용합니다. 최고 속도는 0.8m/s 이하로 제한하고 속도 세분성은 1 또는 2 레벨로 제한해야 합니다.

   고정 위치에 2개의 객체를 배치하여 약 3시간 동안 모델 훈련을 시작합니다. 예제 보상 함수를 사용하여 레이싱할 트랙이나 해당 트랙과 유사한 트랙에서 모델을 훈련합니다. AWS DeepRacer Smile Speedway(중급) 트랙은 단순한 트랙이므로 서밋 레이스 준비에 적합합니다. 동일한 수의 장애물이 있는 동일한 트랙에서 모델을 평가합니다. 총 예상 보상이 어떻게 수렴되는지 주시합니다(수렴할 경우).
2. 보상 함수 매개 변수를 참조하십시오. 보상 함수의 변형을 실험합니다. 장애물을 4개로 늘립니다. 에이전트를 훈련하여 동일한 훈련 시간에 훈련이 수렴되는지 확인합니다. 수렴하지 않으면 보상 함수를 다시 조정하거나 최고 속도를 낮추거나 장애물 수를 줄인 다음 에이전트를 다시 훈련합니다. 더 이상 의미 있는 개선이 없을 때까지 실험을 반복합니다.
3. 이제 무작위 위치에서 장애물을 회피하는 훈련으로 전환합니다. AWS DeepRacer 콘솔의 차고에서 사용할 수 있는 추가 센서를 사용하여 에이전트를 구성해야 합니다. 스테레오 카메라를 사용할 수 있습니다. 또는 LiDAR 장치를 단안 카메라 또는 스테레오 카메라와 결합할 수 있지만 훈련 시간이 길어질 수 있습니다. 훈련을 더 빨리 수렴할 수 있도록 비교적 낮은 최고 속도(예를 들어, 2m/s)로 행동 공간을 설정합니다. 네트워크 아키텍처의 경우 장애물 회피에 충분한 것으로 밝혀진 얕은 신경망을 사용합니다.
4. 단순한 트랙에 장애물 4개를 무작위로 배치하여 4시간 동안 새 에이전트를 훈련합니다. 그런 다음 동일한 트랙에서 모델을 평가하여 무작위로 배치된 장애물이 있는 랩을 완주할 수 있는지 확인합니다. 완주하지 못하는 경우 보상 함수를 조정하고 다른 센서를 사용해 보고 훈련 시간을 늘릴 수 있습니다. 또 한 가지 팁으로, 기존 모델을 복제하여 훈련을 계속하여 이전에 학습한 경험을 활용할 수 있습니다.

5. (선택 사항) 행동 공간에 더 높은 최고 속도를 선택하거나 트랙을 따라 무작위로 더 많은 장애물을 배치합니다. 다양한 센서 조합으로 실험하고 보상 함수 및 하이퍼파라미터 값을 조정합니다. 5계층 CNN 네트워크 토폴로지를 사용하여 실험합니다. 그런 다음 모델을 재훈련하여 훈련 수렴에 어떤 영향을 미치는지 확인합니다.

헤드-투-봇 레이스를 위한 AWS DeepRacer 훈련 조정

장애물 회피 훈련을 마쳤으면 이제 다음 수준의 도전인 헤드-투-봇 레이스를 위한 모델 훈련에 도전할 준비가 되었습니다. 장애물 회피 이벤트와 달리, 헤드-투-봇 레이스는 다른 차량이 주행하는 역동적인 환경에서 벌어집니다. 목표는 AWS DeepRacer 차량이 다른 주행 차량 사이에서 트랙을 벗어나거나 다른 차량과 충돌하지 않고 먼저 결승선에 도착하도록 모델을 훈련하는 것입니다. AWS DeepRacer 콘솔에서는 에이전트가 1~4개의 로봇 차량과 경쟁하도록 하여 헤드-투-봇 레이스 모델을 훈련할 수 있습니다. 일반적으로 더 긴 트랙에 더 많은 장애물이 있어야 합니다.

각 로봇 차량은 일정한 속도로 미리 정의된 경로를 따라 주행합니다. 로봇 차량이 차선을 변경하거나 출발 차선을 유지하도록 할 수 있습니다. 장애물 회피 훈련과 마찬가지로, 로봇 차량이 트랙의 두 차선에 골고루 분포되도록 할 수 있습니다. 콘솔은 트랙에 배치할 수 있는 로봇 차량을 4개로 제한합니다. 트랙에 경쟁하는 차량이 많을수록 학습 에이전트가 다른 차량과 더 다양한 상황에 접할 수 있는 기회를 더 많이 얻을 수 있습니다. 이 방법은 하나의 훈련 작업에서 더 많은 것을 학습하고 에이전트는 더 빨리 훈련됩니다. 그러나 각 훈련이 수렴하는 데 더 오래 걸릴 수 있습니다.

로봇 차량으로 에이전트를 훈련하려면 에이전트 행동 공간의 최고 속도를 로봇 차량의 (일정한) 속도보다 높게 설정해야 훈련 중에 에이전트가 더 많은 추월 기회를 얻을 수 있습니다. 처음에는 에이전트의 최고 속도는 0.8m/s로, 로봇 차량의 이동 속도는 0.4m/s로 설정하는 것이 좋습니다. 로봇이 차선을 변경할 수 있도록 하면 훈련이 더욱 어려워집니다. 에이전트는 같은 차선의 전방에서 이동하는 차량에 충돌하는 것을 방지하는 방법뿐 아니라 다른 차선의 전방에서 이동하는 다른 차량과 충돌하지 않도록 하는 방법을 학습해야 하기 때문입니다. 무작위 간격으로 차선을 변경하도록 로봇 차량을 설정할 수 있습니다. 이 간격은 훈련 작업을 시작하기 전에 지정한 시간 범위(예를 들어, 1~5초)에서 무작위로 선택됩니다. 이 차선 변경 동작은 실제 세계의 헤드-투-봇 레이스 동작과 더 유사하며 훈련된 에이전트가 더 잘 할 수 있습니다. 그러나 모델 훈련이 수렴하는 데 더 오래 걸립니다.

아래에 제안된 단계에 따라 헤드-투-봇 레이스를 위한 훈련을 반복하십시오.

1. AWS DeepRacer 콘솔의 차고에서 스테레오 카메라와 LiDAR 유닛으로 구성된 새로운 교육 에이전트를 빌드하십시오. 로봇 차량에 대해서만 스테레오 카메라를 사용하여 상대적으로 우수한 모델을 훈련할 수 있습니다. LiDAR는 에이전트가 차선을 변경할 때 사각 지대를 줄입니다. 최고 속도를 너무 높게 설정하지 마십시오. 처음에는 1m/s로 시작하는 것이 좋습니다.

2. 헤드-투-봇 레이싱을 위해 훈련하려면 두 대의 봇 차량으로 시작하십시오. 봇의 이동 속도를 에이전트의 최고 속도보다 낮게 설정합니다(예를 들어, 에이전트의 최고 속도가 1m/s인 경우 0.5m/s). 차선 변경 옵션을 비활성화한 다음 방금 생성한 훈련 에이전트를 선택합니다. 보상 함수 예제 중 하나를 사용하거나 최소한으로 필요한 수정을 한 다음 3시간 동안 훈련합니다. 레이싱할 트랙이나 해당 트랙과 유사한 트랙을 사용합니다. AWS DeepRacer Smile Speedway(중급) 트랙은 단순한 트랙이므로 서밋 레이스 준비에 적합합니다. 훈련이 완료되면 동일한 트랙에서 훈련된 모델을 평가합니다.
3. 좀 더 어려운 작업을 위해 두 번째 헤드-투-봇 레이스 모델에 훈련된 모델을 복제합니다. 더 많은 로봇 차량으로 실험을 진행하거나 차선 변경 옵션을 활성화합니다. 2초보다 긴 무작위 간격으로 느린 차선 변경 작업을 시작합니다. 사용자 지정 보상 함수를 실험해 볼 수도 있습니다. 일반적으로 사용자 지정 보상 함수 로직은 다른 차량을 추월하는 것과 차선을 유지하는 것 사이의 균형을 고려하지 않으면 장애물 회피 로직과 비슷할 수 있습니다. 이전 모델이 얼마나 우수한지에 따라 3~6시간을 더 훈련해야 할 수도 있습니다. 모델을 평가하고 모델이 어떤 성능을 보이는지 확인합니다.

AWS DeepRacer 콘솔을 사용하여 AWS DeepRacer 모델을 훈련 및 평가하십시오.

강화 학습 모델을 훈련할 때는 AWS DeepRacer 콘솔을 사용할 수 있습니다. 콘솔에서는 훈련 작업을 생성하거나, 지원되는 프레임워크와 사용 가능한 알고리즘을 선택하거나, 보상 함수를 추가하거나, 훈련 설정을 구성합니다. 또한 시뮬레이터에서 진행되는 훈련을 지켜볼 수도 있습니다. 그 밖에 the section called “첫 번째 모델 훈련”에서 단계별 지침을 찾아보는 것도 가능합니다.

이번 단원에서는 AWS DeepRacer 모델을 훈련하여 평가하는 방법에 대해서 설명합니다. 또한 보상 함수를 생성하고 개선하는 방법과 행동 공간이 모델 성능에 미치는 영향, 그리고 하이퍼파라미터가 훈련 성능에 미치는 영향에 대해서도 살펴보겠습니다. 그 밖에 훈련 모델을 복제하여 훈련 세션을 연장하는 방법, 시뮬레이터를 사용해 훈련 성능을 평가하는 방법, 시뮬레이션에서 실제로 전환하는 데 따른 몇 가지 문제를 해결하는 방법에 대해서도 알아보는 기회가 될 것입니다.

주제

- 보상 함수 생성
- 견고한 모델 훈련을 위한 행동 공간 탐색
- 체계적인 하이퍼파라미터 튜닝
- AWS DeepRacer 교육 작업 진행 상황 조사
- 새로운 훈련 패스를 시작하기 위해 훈련된 모델 복제

- 시뮬레이션에서 AWS DeepRacer 모델 평가
- 실제 환경에 대한 AWS DeepRacer 모델 훈련 최적화

보상 함수 생성

보상 함수는 AWS DeepRacer 차량이 트랙의 한 위치에서 새로운 위치로 이동할 때 즉각적인 피드백(보상 또는 페널티 점수)을 표시합니다. 이 함수의 목적은 차량이 트랙을 따라 이동하면서 사고나 위반 없이 목적지에 빠르게 도달할 수 있도록 하는 데 있습니다. 바람직한 이동일 때는 행동이나 대상 상태에 더욱 높은 점수를 매깁니다. 하지만 잘못되거나 소모적인 이동일 때는 낮은 점수를 매깁니다. AWS DeepRacer 모델을 훈련할 때 유일하게 애플리케이션에게 고유한 부분이 바로 보상 함수입니다.

일반적으로 보상 함수는 마치 인센티브 플랜처럼 설계합니다. 인센티브 전략이 다르면 차량의 동작도 달라질 수 있습니다. 차량 주행 속도를 높이려면 차량이 트랙을 따라 주행할 수 있도록 함수에서 보상을 제공해야 합니다. 차량이 한 바퀴를 완주하는 데 너무 오랜 시간이 걸리거나 트랙에서 벗어날 경우에는 함수가 페널티를 부과해야 합니다. 지그재그 주행 패턴을 피하려면 트랙의 직선 구간에서 조향이 비교적 적은 차량에게 보상을 제공해야 합니다. 차량이 `waypoints`로 측정되는 특정 이정표를 통과할 때는 보상 함수가 양의 점수를 제공하기도 합니다. 이를 통해 차량이 대기하거나 잘못된 방향으로 주행하는 경우를 줄일 수 있습니다. 또한 트랙 조건을 고려하여 보상 함수를 변경할 가능성도 높습니다. 하지만 보상 함수가 환경에 따른 정보를 더욱 많이 고려할수록 훈련된 모델은 과잉 적합으로 인해 보편성이 떨어질 가능성이 더욱 높아집니다. 이때는 행동 공간을 탐색하여 모델의 보편성을 높일 수 있습니다.

인센티브 플랜은 주의해서 생각하지 않으면 역효과라는 의도치 않은 결과로 이어질 수 있습니다. 이는 즉각적인 피드백이 강화 학습을 위한 필요 조건일 뿐 충분 조건은 아니기 때문에 가능한 일입니다. 또한 즉각적인 보상 하나만으로 바람직한 이동 여부를 결정할 수는 없습니다. 예를 들어 임의 위치에서 차량 이동으로 높은 보상 점수를 얻을 수 있습니다. 하지만 이어지는 이동으로 트랙을 벗어나면 낮은 점수를 얻게 됩니다. 이러한 경우에는 차량이 해당 위치에서 높은 점수를 얻은 이동을 회피해야만 합니다. 임의 위치에서 일어나는 모든 미래 이동이 평균적으로 높은 점수를 산출하는 경우에만 다음 위치로 향하는 이동을 바람직하다고 판단해야 합니다. 미래 피드백은 평균 보상 계산에 포함되는 소수의 미래 이동 또는 위치 수를 고려한 비율로 할인됩니다.

보상 함수를 생성할 때는 기본 시나리오에 대한 간단한 함수로 시작하는 것이 좋습니다. 이후 더 많은 행동을 처리할 수 있도록 함수를 개선할 수 있습니다. 이제 몇 가지 간단한 보상 함수를 살펴보겠습니다.

주제

- 간단한 보상 함수 예제

- 보상 함수 개선

간단한 보상 함수 예제

먼저 가장 기본적인 상황을 가정하여 보상 함수를 빌드할 수 있습니다. 가장 기본적인 상황이란 처음부터 끝까지 트랙에서 벗어나지 않고 직선 트랙을 주행하는 것을 말합니다. 이 시나리오에서 보상 함수 로직은 `on_track` 및 `progress`만 고려합니다. 시험 삼아 다음 로직으로 시작할 수 있습니다.

```
def reward_function(params):
    if not params["all_wheels_on_track"]:
        reward = -1
    else if params["progress"] == 1 :
        reward = 10
    return reward
```

이 로직은 트랙에서 벗어났을 때 에이전트에게 페널티를 부과합니다. 또한 결승선까지 주행하면 에이전트에게 보상을 제공합니다. 이는 지정된 목표를 달성하는 데 합리적입니다. 그러나 에이전트가 트랙에서 역주행하는 것을 포함해 출발선과 결승선 사이를 자유롭게 돌아 다닙니다. 훈련을 완료하는 데 시간이 오래 걸릴 뿐만 아니라 훈련된 모델을 실제 차량에 배포하면 주행 효율성이 떨어질 수도 있습니다.

실제로, 훈련 과정에서 하나씩 배워갈 수 있을 때 에이전트는 보다 효율적으로 배울 수 있습니다. 이는 보상 함수가 트랙을 따라 단계별로 좀 더 작은 보상을 제공해야 함을 의미합니다. 에이전트가 직선 트랙을 주행하도록 하기 위해 다음과 같이 보상 함수를 개선할 수 있습니다.

```
def reward_function(params):
    if not params["all_wheels_on_track"]:
        reward = -1
    else:
        reward = params["progress"]
    return reward
```

이 함수를 사용하면 에이전트는 결승선에 더 가까이 갈수록 더 큰 보상을 얻습니다. 따라서 역주행하는 비생산적인 시도가 줄어들거나 없어집니다. 일반적으로 우리는 보상 함수가 작용 영역 전체에서 보상을 보다 균일하게 분산시키길 원합니다. 효과적인 보상 함수를 생성하는 일은 어려울 수 있습니다. 따라서 처음에는 간단한 함수부터 시작하여 점차 함수를 개선하는 것이 바람직합니다. 체계적인 실험을 통해 함수를 더욱 견고하게, 그리고 효율적으로 개선할 수 있습니다.

보상 함수 개선

간단한 직선 트랙에서 AWS DeepRacer 모델을 성공적으로 훈련했으면 AWS DeepRacer 차량(가상 또는 물리) 이 트랙을 벗어나지 않고 제대로 주행할 수 있습니다. 차량이 순환 트랙에서 주행하도록 두면 트랙을 벗어납니다. 보상 함수가 트랙을 따라 회전해야 하는 작업을 무시합니다.

차량이 이러한 행동을 처리할 수 있으려면 보상 함수를 개선해야 합니다. 보상 함수는 에이전트가 허용되는 회전을 하면 보상을 제공하고, 에이전트가 잘못된 회전을 하면 페널티를 부과해야 합니다. 그런 다음에는 다른 훈련 단계를 시작할 준비가 된 것입니다. 이전 교육을 활용하려면 이전에 훈련된 모델을 복제하여 새 교육을 시작함으로써 이전에 학습한 지식을 전달할 수 있습니다. 이전에 학습한 패턴에 따라 점점 더 복잡한 환경에서 주행하도록 AWS DeepRacer 차량을 훈련하기 위해 보상 함수에 기능을 점진적으로 추가할 수 있습니다.

고급 보상 함수는 다음 예를 참조하십시오.

- the section called “예제 1: 타임 트라이얼 시 중앙선 따르기”
- the section called “예제 2: 타임 트라이얼에서 두 경계선 사이를 유지”
- the section called “예제 3: 타임 트라이얼에서 지그재그 주행 방지”
- the section called “예제 4: 정지 장애물 또는 이동 차량에 충돌하지 않고 한 차선을 유지”

견고한 모델 훈련을 위한 행동 공간 탐색

일반적으로 모델을 다양한 환경에 적용하려면 모델을 최대한 견고하게 훈련해야 합니다. 여기에서 견고한 모델이란 광범위한 트랙 형상 및 조건에 적용할 수 있는 모델을 말합니다. 견고한 모델이라고 해서 “스마트”한 것은 아닙니다. 보상 함수는 환경에 따른 명시적 정보를 추가할 능력이 없기 때문입니다. 오히려 모델은 훈련된 환경과 비슷한 환경일 때만 적용될 가능성이 높습니다.

환경에 따른 정보를 보상 함수에 명시적으로 추가하는 일은 피처 엔지니어링 분야입니다. 피처 엔지니어링은 훈련 시간을 줄이는 데 효과적일 뿐만 아니라 특정 환경에 맞춘 솔루션에서도 유용할 수 있습니다. 그렇더라도 일반적으로 적용되는 모델을 훈련하려면 피처 엔지니어링을 많이 시도해서도 안 됩니다.

예를 들어 원형 트랙에서 모델을 훈련하면서 이러한 지오메트리 속성을 보상 함수에 명시적으로 추가했다면 원형이 아닌 트랙에 적용할 수 있는 모델은 기대하기 어렵습니다.

이때 보상 함수를 최대한 단순하게 유지하면서 동시에 모델을 최대한 견고하게 훈련하려면 어떻게 해야 할까요? 첫째, 에이전트의 모든 행동이 일어나는 행동 공간을 탐색하는 방법이 있습니다. 둘째, 기본 훈련 알고리즘의 하이퍼파라미터를 가지고 실험하는 방법이 있습니다. 두 방법을 모두 사용하는 경우

가 많습니다. 여기에서는 행동 공간을 탐색하여 AWS DeepRacer 차량 모델을 견고하게 훈련하는 방법을 중심으로 설명하겠습니다.

AWS DeepRacer 모델을 훈련할 때 행동(a)은 속도(초당 t 미터)와 조향 각도(s도)의 조합으로 나타납니다. 에이전트의 속도와 조향 각도 범위는 에이전트의 행동 공간에 따라 다릅니다. $(v_1, .., v_n)$에서 속도 수가 m이고, $(s_1, .., s_m)$에서 조향 각도 수가 n인 행동 공간이 있다고 가정할 경우 행동 공간에서 일어날 수 있는 행동은 m*n입니다.

```
a1:            (v1, s1)
...
an:            (v1, sn)

...
a(i-1)*n+j:   (vi, sj)
...

a(m-1)*n+1:   (vm, s1)
...
am*n:          (vm, sn)
```

(v_i, s_j)의 실제 값은 v_{max}와 $|s_{max}|$의 범위에 따라 다르며, 균일하게 분산되지도 않습니다.

AWS DeepRacer 모델을 훈련하거나 반복할 때마다 먼저 n, m, v_{max}, 및 $|s_{max}|$를 지정하거나, 혹은 기본값을 사용하는 데 동의해야 합니다. 이 선택에 따라 AWS DeepRacer 서비스는 에이전트가 훈련에서 선택할 수 있는 행동을 생성합니다. 이렇게 생성된 행동은 행동 공간에서 균일하게 분산되지 않습니다.

일반적으로 행동 수가 많고, 행동 범위가 클수록 에이전트가 회전 각도 또는 방향이 불규칙한 곡선 트랙 등 다양한 트랙 조건에 대응할 수 있는 가능성 또는 옵션도 늘어납니다. 에이전트에 사용할 수 있는 옵션이 늘어나면 트랙 변화에 즉시 대처할 수 있는 능력도 커집니다. 결과적으로 단순한 보상 함수를 사용할 때조차 훈련된 모델을 더욱 광범위하게 적용할 수 있는 효과를 기대할 수 있습니다.

예를 들어 에이전트는 조잡한 행동 공간에서도 적은 수의 속도와 조향 각도를 사용해 직선 구간에 대처하는 방법을 빠르게 학습할 수 있습니다. 곡선 구간에서는 행동 공간이 이렇게 조잡하면 에이전트가 회전하면서 오버슈트를 일으켜 트랙에서 벗어날 가능성이 높습니다. 이는 속도 또는 조향 각도를 조정하기 위해 사용할 수 있는 옵션이 충분하지 않기 때문입니다. 이때는 속도 수 또는 조향 각도 수를, 아니면 둘 다 늘려서 에이전트가 트랙을 벗어나지 않으면서 곡선 구간에 대처할 수 있는 능력을 높여야 합니다. 마찬가지로 에이전트가 지그재그 방식으로 이동할 경우에는 조향 각도 수를 늘려서 임의 단계에서 지나친 방향 전환을 줄일 수 있습니다.

행동 공간이 너무 크면 훈련 성능이 떨어질 수 있습니다. 행동 공간을 탐색하는 데 오랜 시간이 걸리기 때문입니다. 따라서 모델을 일반적으로 적용할 수 있는 이점과 훈련 성능 요건 사이에 균형을 안정적으로 유지해야 합니다. 이러한 최적화는 체계적인 실험이 수반됩니다.

체계적인 하이퍼파라미터 튜닝

모델 성능을 개선하려면 훈련 프로세스의 효과를 높이는 방법이 있습니다. 예를 들어 견고한 모델을 얻으려면 훈련할 때 행동 공간에 다소 균일하게 분산된 샘플을 에이전트에게 제공해야 합니다. 이를 위해서는 탐색과 활용이 충분하게 혼합되어야 합니다. 여기에 영향을 미치는 변수로는 사용되는 훈련 데이터의 크기(`number of episodes between each training` 및 `batch size`)와 에이전트의 학습 속도(`learning rate`), 그리고 탐색 구간(`entropy`)이 있습니다. 또한 실용적인 훈련을 위해 학습 프로세스의 속도를 높여야 할 수도 있습니다. 여기에 영향을 미치는 변수로는 `learning rate`, `batch size`, `number of epochs` 및 `discount factor`가 있습니다.

훈련 프로세스에 영향을 미치는 변수는 훈련 하이퍼파라미터라고 불립니다. 이러한 알고리즘 속성은 기본 모델의 속성이 아닙니다. 아쉽지만 하이퍼파라미터는 기본적으로 실증적입니다. 최적의 값은 실제로 알려져 있지 않기 때문에 체계적인 실험을 통해서 최적의 값을 도출해야 합니다.

조정을 통해 AWS DeepRacer 모델의 훈련 성능을 튜닝할 수 있는 하이퍼파라미터에 대해 얘기하기 전에 먼저 다음 용어 정의에 대해 알아보겠습니다.

데이터 포인트

경험이라고도 불리는 데이터 포인트는 (s,a,r,s’)와 같은 튜플을 말합니다. 여기에서 s는 카메라에서 수집되는 관측 결과(상태)를, a는 차량 행동을, r은 차량 행동에 따라 예상되는 보상을, 그리고 s’는 행동 이후 새로운 관측 결과를 의미합니다.

에피소드

에피소드는 차량이 임의 출발점에서 시작하여 트랙을 완주하거나 트랙에서 벗어나 주행을 종료할 때까지 걸리는 기간을 말합니다. 이러한 에피소드를 통해 연속된 경험이 구체화됩니다. 에피소드에 따라 길이가 달라질 수 있습니다.

경험 버퍼

경험 버퍼는 훈련 과정에서 다양한 길이로 고정된 다수의 에피소드를 지나면서 수집 및 정리된 여러 데이터 포인트로 구성됩니다. AWS DeepRacer에서는 AWS DeepRacer 차량에 장착된 카메라를 통해 수집된 이미지와 차량의 행동이 여기에 해당하며, 기본(정책 및 가치) 신경망을 업데이트할 목적으로 입력 값을 가져오는 소스의 역할을 합니다.

배치

배치는 경험이 순서대로 나열된 목록으로 정책망 가중치를 업데이트하는 데 사용됩니다. 여기에서 경험이란 일정 시간을 지나면서 실시한 시뮬레이션의 일부를 말합니다. 배치는 경험 버퍼의 하위 집합입니다.

훈련 데이터

훈련 데이터는 경험 버퍼에서 무작위로 샘플링되는 배치 집합으로서 정책망 가중치를 훈련하는 데 사용됩니다.

알고리즘 하이퍼파라미터와 하이퍼파라미터의 효과

하이퍼파라미터	설명
경사 하강 배치 크기	경험 버퍼에서 무작위로 샘플링된 최근 차량 경험의 수로서 기본 딥 러닝 신경망 가중치를 업데이트하는 데 사용됩니다. 무작위 샘플링은 입력 데이터에 고유한 상관관계를 줄이는 데 효과적입니다. 신경망 가중치를 더욱 안정적으로 업데이트하려면 배치 크기를 높이십시오. 단, 훈련이 길어지거나 느려질 수 있다는 점은 알고 있어야 합니다. 필수 예 유효값 양의 정수 (32, 64, 128, 256, 512) 기본값 64
에포크 수	경사 하강 과정에서 신경망 가중치를 업데이트할 목적으로 훈련 데이터를 통하는 패스(pass) 수입니다. 훈련 데이터는 경험 버퍼의 무작위 샘플에 해당합니다. 더욱 안정적인 업데이트를 원한다면 에포크 수를 늘리십시오. 단, 훈련이 느려질 수 있습니다. 배치 크기가 작을 때는 에포크도 줄여서 사용할 수 있습니다. 필수 아니요

하이퍼파라미터	설명
	유효값 [3 - 10] 사이의 음의 정수 기본값 3
학습 속도	업데이트할 때마다 새로운 가중치의 일부는 경사 하강(또는 상승) 기여도에서 가져오고, 나머지는 기존 가중치 값에서 가져올 수 있습니다. 학습 속도는 신경망 가중치에 대한 경사 하강(또는 상승) 업데이트의 기여도를 제어합니다. 학습 속도를 높여서 경사 하강 기여도를 늘리면 훈련 속도가 빨라지지만 학습 속도가 너무 높을 경우 예상하는 보상이 수렴되지 않을 수도 있다는 점을 알고 있어야 합니다. 필수 아니요 유효값 0.00000001 (또는 10^{-8}) 및 0.001(또는 10^{-3}) 사이의 실수 기본값 0.0003

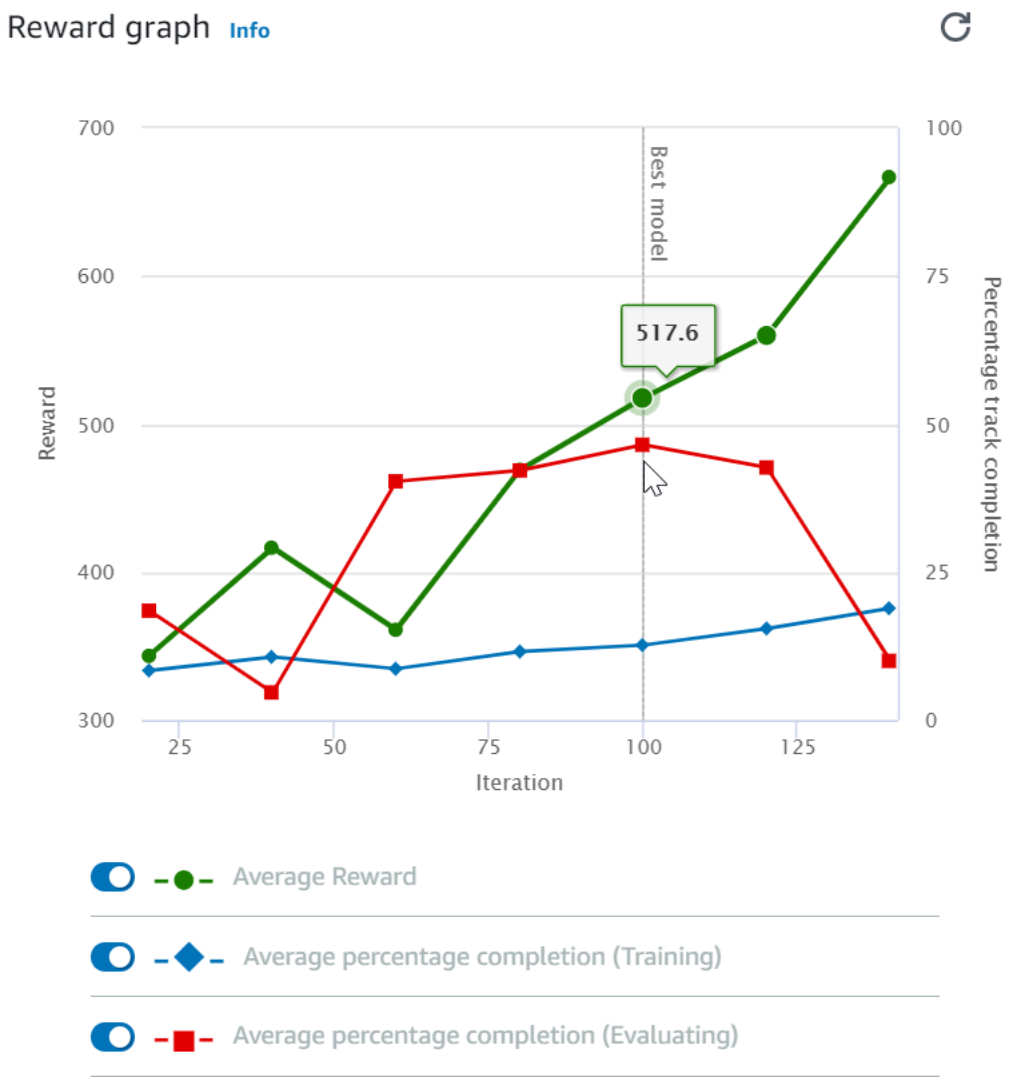
하이퍼파라미터	설명
Entropy	정책 분포에 무작위성을 추가할 시점을 결정할 때 사용되는 불확실성입니다. 불확실성이 추가되면 AWS DeepRacer 차량이 행동 공간을 더욱 넓게 탐색하는 데 유용합니다. 반면, 엔트로피 값이 커지면 차량이 행동 공간을 더욱 철저하게 탐색할 수 있습니다. 필수 아니요 유효값 0과 1 사이의 실수 기본값 0.01
할인 계수	할인 계수는 예상되는 보상에 대한 미래 보상의 기여도를 지정합니다. 할인 계수 값이 클수록 차량이 이동 시 고려하는 기여도가 더욱 멀어지기 때문에 훈련 속도가 느려집니다. 예를 들어 할인 계수가 0.9이면 차량이 이후 10 단계의 보상을 고려하여 이동합니다. 할인 계수가 0.999이면 차량이 이후 1,000 단계의 보상을 고려하여 이동합니다. 권장되는 할인 계수 값은 0.99, 0.999 및 0.9999입니다. 필수 아니요 유효값 0과 1 사이의 실수 기본값 0.999

하이퍼파라미터	설명
손실 유형	신경망 가중치를 업데이트하는 데 사용되는 목적 함수의 한 가지 유형입니다. 훌륭한 훈련 알고리즘은 에이전트의 전략에 증분 변경을 적용하여 임의 작업을 수행하는 단계에서 보상을 늘리기 위한 전략적 작업을 수행하는 단계로 점진적으로 나아가도록 해야 합니다. 그러나 너무 크게 변경되면 훈련이 불안정해지고 결국 에이전트가 학습하지 못할 수 있습니다. Huber 손실 및 평균 제곱근 오차 손실 유형은 소규모 업데이트에 대해 유사하게 작동합니다. 그러나 업데이트의 규모가 커지면 Huber 손실은 평균 제곱근 오차 손실에 비해 더 작게 증분됩니다. 수렴 문제가 있는 경우에는 Huber 손실 유형을 사용합니다. 수렴이 적절하고 더 빨리 훈련시키고자 하는 경우에는 평균 제곱근 오차 손실 유형을 사용합니다. 필수 아니요 유효값 (Huber 손실, 평균 제곱근 오차 손실) 기본값 Huber 손실

하이퍼파라미터	설명
각 정책 업데이트 반복 간 경험 에피소드의 수	정책망 가중치를 학습할 목적으로 훈련 데이터를 가져올 때 사용되는 경험 버퍼의 크기입니다. 경험 에피소드는 에이전트가 임의 출발점에서 시작하여 트랙을 완주하거나 트랙에서 벗어나 주행을 종료할 때까지 걸리는 기간을 말합니다. 이 에피소드는 연속된 경험으로 구성됩니다. 에피소드에 따라 길이가 달라질 수 있습니다. 간단한 강화 학습 문제일 때는 작은 크기의 경험 버퍼로도 충분할 수 있어서 학습 속도가 빠릅니다. 하지만 국소 최대값이 많아서 강화 학습 문제가 더욱 복잡해지면 경험 버퍼의 크기가 커야만 비상관 데이터 포인트를 더욱 많이 제공할 수 있습니다. 이 경우 훈련 속도가 느려지지만 더욱 안정적입니다. 권장 값은 10, 20 및 40입니다. 필수 아니요 유효값 5와 100 사이의 정수입니다. 기본값 20

AWS DeepRacer 교육 작업 진행 상황 조사

훈련 작업을 시작하고 에피소드당 보상 및 트랙 완료의 훈련 지표를 조사하여 모델의 훈련 작업 성능을 알아낼 수 있습니다. AWS DeepRacer 콘솔에서 지표는 다음 그림처럼 보상 그래프에 나타납니다.



에피소드당 획득한 보상, 반복당 평균 보상, 에피소드당 진행 상황, 반복당 평균 진행 상황 또는 이들의 조합을 볼 수 있도록 선택할 수 있습니다. 이렇게 하려면 보상 그래프 하단에서 보상(에피소드, 평균) 또는 진행률(에피소드, 평균) 스위치를 전환하십시오. 에피소드당 보상과 진행 상황은 다른 색깔의 산점도로 표시됩니다. 평균 보상과 트랙 완료는 선표본점으로 표시되며 첫 번째 반복 이후 시작됩니다.

보상의 범위는 그래프의 왼쪽에 나타나며 진행 상황의 범위(0-100)은 오른쪽에 나타납니다. 훈련 지표의 정확한 값을 읽으려면, 그래프의 데이터 포인트 근처로 마우스를 옮깁니다.

훈련이 진행 중일 때는 그래프가 10초마다 자동으로 업데이트됩니다. 새로 고침 버튼을 선택하여 지표 디스플레이를 수동으로 업데이트할 수 있습니다.

평균 보상과 트랙 완료가 수렴하는 경향을 보이면 훈련 작업이 잘 된 것입니다. 특히 에피소드당 진행 상황이 계속해서 100%에 도달하고 보상이 수준을 벗어나면 모델의 수렴 가능성이 높습니다. 그렇지 않으면, 모델을 복제하고 다시 훈련합니다.

새로운 훈련 패스를 시작하기 위해 훈련된 모델 복제

이전에 훈련된 모델을 복제하여 새로운 훈련을 시작하면 훈련 효율성을 개선할 수 있습니다. 이를 위해서는 이전에 학습된 지식을 사용할 수 있도록 하이퍼파라미터를 수정합니다.

이번 단원에서는 AWS DeepRacer 콘솔을 사용하여 훈련된 모델을 복제하는 방법에 대해서 알아보겠습니다.

AWS DeepRacer 콘솔을 사용하여 강화 학습 모델 훈련을 반복하는 방법

1. 아직 로그인하지 않았다면 AWS DeepRacer 콘솔에 로그인합니다.
2. 모델 페이지에서 학습된 모델을 선택한 다음 작업 드롭다운 메뉴 목록에서 복제를 선택합니다.
3. 모델 세부 정보에서 다음 작업을 수행합니다.
   a. 복제된 모델의 이름을 생성하지 않으려면 모델 이름에 `RL_model_1`을 입력합니다.
   b. 선택적으로 모델 설명 - 옵션에 복제한 모델에 대한 설명을 입력합니다.
4. 환경 시뮬레이션에서 다른 트랙 옵션을 선택합니다.
5. 보상 함수에서 사용 가능한 보상 함수 예제 하나를 선택합니다. 보상 함수를 수정합니다. 예를 들어 조향 각도를 수정합니다.
6. 알고리즘 설정을 확장하여 다른 옵션들도 바꿔봅니다. 예를 들어 Gradient descent batch size(경사 하강 배치 크기) 값을 32에서 64로 변경하거나, 혹은 학습 속도를 높여 훈련 속도를 높입니다.
7. 중지 조건의 여러 가지 옵션을 사용하여 실험합니다.
8. 그런 다음 훈련 시작을 선택하고 새로운 훈련을 시작합니다.

일반적으로 견고한 기계 학습 모델을 훈련하는 것과 마찬가지로, 최고의 솔루션을 찾기 위해서는 체계적인 실험을 수행해야 합니다.

시뮬레이션에서 AWS DeepRacer 모델 평가

모델 평가는 훈련된 모델의 성능을 테스트하는 것입니다. AWS DeepRacer에서 표준 성능 지표는 3바퀴를 연속해서 완주하는 데 걸리는 평균 시간입니다. 임의의 두 모델을 가지고 이러한 지표를 사용했을 때 에이전트 하나가 동일한 트랙에서 더욱 빠르게 완주한다면 나머지 하나보다 더욱 성능이 높은 것입니다.

일반적으로 모델 평가는 다음과 같은 작업이 수반됩니다.

1. 평가 작업을 구성하여 시작합니다.

2. 작업 실행 중 평가 진행 과정 관찰. 이 작업은 AWS DeepRacer 시뮬레이터에서 실행할 수 있습니다.
3. 평가 작업 완료 후 평가 요약 검사. 평가 작업은 진행 도중 언제든지 종료할 수 있습니다.

> **Note**
>
> 평가 시간은 선택한 기준에 따라 달라집니다. 모델이 평가 기준을 충족하지 않는 경우 최대 20분에 도달할 때까지 평가가 계속 진행됩니다.

4. 선택 사항이지만 평가 결과를 관련 AWS DeepRacer 리더보드에 제출합니다. 리더보드의 순위를 통해 자신의 모델 성능을 다른 참가자와 비교해 알아볼 수 있습니다.

물리적인 트랙에서 AWS DeepRacer 차량을 주행하면서 AWS DeepRacer 모델을 테스트할 때는 차량 작동 단원을 참조하십시오.

실제 환경에 대한 AWS DeepRacer 모델 훈련 최적화

훈련된 모델의 실제 성능에 영향을 미치는 요인은 행동 공간, 보상 함수, 훈련에 사용되는 하이퍼파라미터, 차량 보정, 실제 트랙 조건 등 매우 많습니다. 그 밖에도 시뮬레이션은 실제 환경에 대한 (종종 대략적인) 근사값일 뿐입니다. 모델을 시뮬레이션에서 훈련한 후 실제로 적용하여 만족스러운 성능을 얻기 어려운 이유도 바로 여기에 있습니다.

모델을 훈련하여 실제 환경에서도 견고한 성능을 얻으려면 보상 함수, 행동 공간 및 하이퍼파라미터 탐색을 비롯해 시뮬레이션 평가, 실제 환경 테스트를 수없이 반복해야 하는 경우가 많습니다. 특히 마지막 단계에서는 시뮬레이션에서 실제 환경으로(sim2real) 전환하는 작업이 수반되어 번거롭게 느낄 수도 있습니다.

다음은 sim2real 과제를 해결할 때 주의해야 할 사항입니다.

- 차량이 적절하게 보정되어 있는지 확인하십시오.

  보정 확인은 시뮬레이션 환경은 실제 환경을 부분적으로 나타낼 가능성이 매우 높기 때문에 중요한 절차입니다. 더욱이 에이전트는 각 단계마다 카메라의 이미지에서 수집되는 현재 트랙 조건을 기준으로 행동합니다. 따라서 충분히 멀리 보지 못해서 빠른 속도로 경로를 계획할 수 없습니다. 이러한 문제를 해결하기 위해 시뮬레이션은 속도와 조향 각도를 제한합니다. 또한 훈련된 모델이 실제 환경에서 유효하게 적용되려면 차량이 실제 환경과 기타 시뮬레이션 설정에 맞춰 적절하게 보정되어야 합니다. 차량 보정에 대한 자세한 내용은 the section called “차량 보정” 단원을 참조하십시오.
- 기본 모델을 사용해 차량을 먼저 테스트하십시오.

AWS DeepRacer 차량은 사전 훈련된 모델이 추론 엔진에 탑재되어 제공됩니다. 자신의 모델을 실제 환경에서 테스트하려면 먼저 차량이 기본 모델을 사용했을 때 타당한 성능을 발휘하는지 확인하십시오. 그렇지 않다면 물리적 트랙 환경을 점검해야 합니다. 물리적 트랙이 잘못 빌드되어 있다면 성능이 떨어질 가능성이 높습니다. 이때는 테스트를 시작하거나 재개하기 전에 트랙을 재구성하거나 수정해야 합니다.

> **Note**
>
> AWS DeepRacer 차량을 구동할 때는 보상 함수를 간접적으로 호출하지 않고 훈련된 정책망에 따라 행동이 추론됩니다.

- 시뮬레이션에서 모델 유효성을 확인하십시오.

  모델이 실제 환경에서 성능을 발휘하지 못한다면 모델 또는 트랙에 문제가 있을 가능성이 높습니다. 근본 원인을 규명하려면 먼저 시뮬레이션에서 모델을 평가하여 시뮬레이션 에이전트가 트랙을 벗어나지 않고 1바퀴 이상 완주하는지 확인해야 합니다. 평가할 때는 시뮬레이터에서 에이전트의 궤적을 관측하면서 보상 수렴을 검사합니다. 시뮬레이션 에이전트가 주춤거리지 않고 한 바퀴를 완주하면서 보상이 최대값에 도달하면 모델에는 문제가 없을 가능성이 높습니다.

- 모델을 과도하게 훈련하지 마십시오.

  모델이 시뮬레이션에서 일관적으로 트랙을 완주하는 데도 훈련을 계속하면 모델 과잉 적합의 원인이 됩니다. 과도하게 훈련된 모델은 실제 환경에서 성능을 발휘하지 못합니다. 시뮬레이션 트랙과 실제 환경 사이에서 편차가 조금만 발생해도 대처하지 못하기 때문입니다.

- 반복할 때마다 여러 모델을 사용하십시오.

  일반적인 훈련 세션에서는 과소 적합과 과잉 적합 사이에서 다양한 모델이 생성됩니다. 적합한 모델에 대한 사전 기준이 없기 때문에 에이전트가 시뮬레이터에서 한 바퀴 완주하는 시점부터 두 바퀴 이상 일관적으로 순환하는 시점 사이에서 모델 후보를 몇 개 선정해야 합니다.

- 처음에는 저속으로 시작하여 조금씩 주행 속도를 높이면서 테스트하십시오.

  차량에 배포할 모델을 테스트할 때는 처음에 최저 속도 값으로 시작합니다. 예를 들어 테스트 속도 제한을 훈련 속도 제한의 10% 미만으로 설정할 수 있습니다. 그런 다음 차량이 이동하기 시작할 때까지 테스트 속도를 조금씩 높입니다. 디바이스 제어 콘솔에서 차량을 보정할 때 테스트 속도 제한도 설정합니다. 예를 들어, 차량이 너무 빠를 경우, 즉 속도가 시뮬레이터의 훈련 속도를 초과하면 모델이 실제 트랙에서 성능을 제대로 발휘하지 못할 가능성이 높습니다.

- 여러 출발점에서 차량을 사용해 모델을 훈련하십시오.

모델은 시뮬레이션에서 특정 경로를 학습하기 때문에 트랙에서 자신의 위치에 민감할 수 있습니다. 따라서 트랙 경계 내 여러 위치(왼쪽부터 중앙을 거쳐 오른쪽까지)에서 차량 테스트를 시작하여 모델이 어느 위치에서 성능을 발휘하는지 살펴보아야 합니다. 대부분 모델은 차량을 흰색 라인 중 하나의 한쪽 측면에 가깝게 유지하려는 경향이 있습니다. 차량 경로를 분석하려면 차량의 위치(x, y)를 시뮬레이션부터 단계별로 구분하여 차량이 실제 환경에서 이동할 가능성이 높은 경로를 식별하는 것이 좋습니다.

- 처음에는 직선 트랙으로 테스트하십시오.

  직선 트랙은 곡선 트랙과 비교해 탐색하기 훨씬 쉽습니다. 직선 트랙부터 테스트를 시작하면 성능이 저조한 모델부터 빠르게 제거하는 데 유용합니다. 차량이 직선 트랙을 따라 거의 주행하지 못하면 곡선 트랙에서도 성능을 발휘하지 못합니다.

- 차량이 여러 행동 중 한 가지 유형만 보이는 동작이 있는지 주의하여 지켜보십시오.

  예를 들어, 조향이 왼쪽으로만 치우치는 등 차량이 여러 행동 중 한 가지 유형만 보일 때는 모델이 과잉 적합이거나 과소 적합일 가능성이 높습니다. 임의의 모델 파라미터로 훈련을 지나치게 많이 반복하면 모델에 과잉 적합이 일어날 수 있습니다. 반대로 반복이 너무 적으면 과소 적합이 일어날 수 있습니다.

- 따라서 차량이 트랙 경계를 따라 경로를 수정할 수 있는지 지켜보십시오.

  차량이 트랙 경계에 근접할 때 경로를 스스로 수정할 수 있다면 우수한 모델이라고 할 수 있습니다. 훈련이 잘 된 모델은 대부분 경로를 스스로 수정할 수 있습니다. 차량이 두 트랙 경계에서 스스로 경로를 수정할 때 더욱 견고하고 품질이 높은 모델이라고 할 수 있습니다.

- 불규칙한 차량 동작 유무를 주의하여 지켜보십시오.

  정책 모델은 임의 상태에서 행동일 보일 수 있는 확률 분포를 나타냅니다. 효과적으로 훈련된 모델이 추론 엔진에 탑재되어 있다면 차량은 모델 지시에 따라 한 번에 한 단계씩 가장 유망한 행동을 선택합니다. 행동 확률이 균일하게 분포한다면 차량이 동일하거나 거의 비슷한 확률의 행동을 보일 수 있습니다. 이는 불규칙한 주행 동작의 원인이 됩니다. 예를 들어 차량이 직선 구간을 주행하는 경우가 많고(예를 들어, 주행 시간의 절반 이상) 다른 때는 불필요한 회전을 한다면 모델이 과소 적합이거나 과잉 적합입니다.

- 차량이 여러 회전 중 한 가지 유형(왼쪽 또는 오른쪽)만 보이는지 주의하여 지켜보십시오.

  차량이 지나치게 왼쪽으로 회전하지만 오른쪽으로 조향하지 못하는 경우, 혹은 마찬가지로 차량이 지나치게 오른쪽으로 회전하지만 왼쪽으로 조향하지 못한다면 차량의 조향 각도를 주의하여 보정하거나 재보정해야 합니다. 또한 테스트 과정에서 물리적 설정에 가까운 값으로 훈련된 모델을 사용하는 것도 좋은 방법입니다.

- 차량이 갑자기 회전하거나 트랙에서 벗어나는지 주의하여 지켜보십시오.

  차량이 대부분 경로를 따라 정확하게 이동하다 갑자기 트랙을 이탈한다면 주변 환경에 주의를 분산시키는 요인이 있을 가능성이 높습니다. 가장 자주 발생하는 분산 요인으로는 예상하지 못하거나 의도하지 않은 빛 반사가 있습니다. 이때는 트랙 주위에 장벽을 설치하거나 그 밖에 밝은 빛을 줄일 수 있는 수단을 사용하십시오.

AWS DeepRacer 보상 함수 참조

다음은 AWS DeepRacer 보상 함수의 기술 참조입니다.

주제

- AWS DeepRacer 보상 함수의 입력 파라미터
- AWS DeepRacer 보상 함수 예제

AWS DeepRacer 보상 함수의 입력 파라미터

AWS DeepRacer 보상 함수는 딕셔너리 객체를 입력 값으로 갖습니다.

```
def reward_function(params) :

    reward = ...

    return float(reward)
```

`params` 딕셔너리 객체에는 다음과 같은 키-값 페어가 저장됩니다.

```
{
    "all_wheels_on_track": Boolean,        # flag to indicate if the agent is on the
 track
    "x": float,                            # agent's x-coordinate in meters
    "y": float,                            # agent's y-coordinate in meters
    "closest_objects": [int, int],         # zero-based indices of the two closest
 objects to the agent's current position of (x, y).
    "closest_waypoints": [int, int],       # indices of the two nearest waypoints.
    "distance_from_center": float,         # distance in meters from the track center
    "is_crashed": Boolean,                 # Boolean flag to indicate whether the agent
 has crashed.
```

```
    "is_left_of_center": Boolean,          # Flag to indicate if the agent is on the
 left side to the track center or not.
    "is_offtrack": Boolean,                # Boolean flag to indicate whether the agent
 has gone off track.
    "is_reversed": Boolean,                # flag to indicate if the agent is driving
 clockwise (True) or counter clockwise (False).
    "heading": float,                      # agent's yaw in degrees
    "objects_distance": [float, ],         # list of the objects' distances in meters
 between 0 and track_length in relation to the starting line.
    "objects_heading": [float, ],          # list of the objects' headings in degrees
 between -180 and 180.
    "objects_left_of_center": [Boolean, ], # list of Boolean flags indicating whether
 elements' objects are left of the center (True) or not (False).
    "objects_location": [(float, float),], # list of object locations [(x,y), ...].
    "objects_speed": [float, ],            # list of the objects' speeds in meters per
 second.
    "progress": float,                     # percentage of track completed
    "speed": float,                        # agent's speed in meters per second (m/s)
    "steering_angle": float,               # agent's steering angle in degrees
    "steps": int,                          # number steps completed
    "track_length": float,                 # track length in meters.
    "track_width": float,                  # width of the track
    "waypoints": [(float, float), ]        # list of (x,y) as milestones along the
 track center

}
```

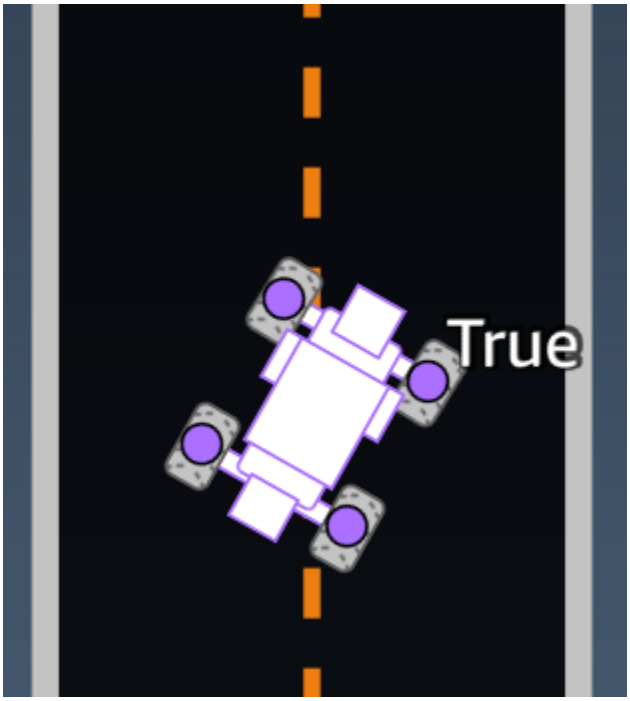
입력 파라미터에 대해 자세한 기술 참조는 다음과 같습니다.

all_wheels_on_track

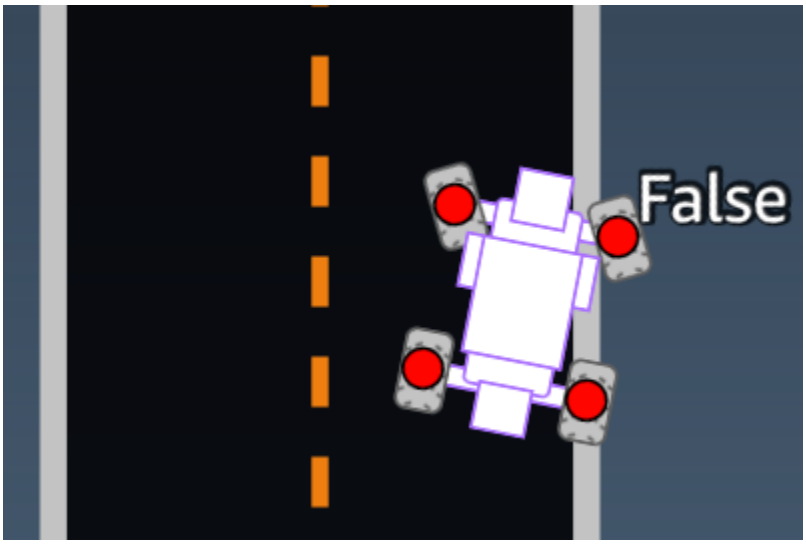
유형: `Boolean`

범위: `(True:False)`

에이전트의 트랙 주행 또는 트랙 이탈을 나타내는 `Boolean` 플래그입니다. 바퀴 하나라도 트랙 경계를 벗어나면 트랙 이탈(`False`)입니다. 바퀴가 모두 트랙 경계 사이에 있으면 트랙 주행(`True`)입니다. 다음 그림은 에이전트가 트랙을 따라 주행하는 것을 나타냅니다.



다음 그림은 에이전트가 트랙을 이탈한 것을 나타냅니다.



예제: *all_wheels_on_track* 파라미터를 사용하는 보상 함수

```
def reward_function(params):
    ################################################################################
    '''
    Example of using all_wheels_on_track and speed
    '''

    # Read input variables
```

```
    all_wheels_on_track = params['all_wheels_on_track']
    speed = params['speed']

    # Set the speed threshold based your action space
    SPEED_THRESHOLD = 1.0

    if not all_wheels_on_track:
        # Penalize if the car goes off track
        reward = 1e-3
    elif speed < SPEED_THRESHOLD:
        # Penalize if the car goes too slow
        reward = 0.5
    else:
        # High reward if the car stays on track and goes fast
        reward = 1.0

    return float(reward)
```

closest_waypoints

유형: `[int, int]`

범위: `[(0:Max-1),(1:Max-1)]`

에이전트의 현재 위치인 `(x, y)`에 가장 가깝게 인접한 두 `waypoint`의 제로 기반 인덱스입니다. 거리는 에이전트 중앙에서 유클리트(Eudlidean) 거리로 측정됩니다. 첫 번째 요소는 에이전트 뒤에서 가장 가까운 중간 지점을 나타내고, 두 번째 요소는 에이전트 앞에서 가장 가까운 중간 지점을 나타냅니다. `Max`는 중간 지점 목록의 길이입니다. 중간 지점에서 나타난 그림에서 `closest_waypoints`는 `[16, 17]`입니다.

예: `closest_waypoints` 파라미터를 사용하는 보상 함수

다음 보상 함수 예제는 `waypoints`, `closest_waypoints` 및 `heading`을 사용해 즉각적인 보상을 계산하는 방법을 나타낸 것입니다.

AWS DeepRacer는 수학, 무작위, NumPy, SciPy, Shapely 등의 라이브러리를 지원합니다. 이를 사용하려면 함수 정의 `def function_name(parameters)` 위에 가져오기 문 `import` *`supported library`*을 추가하십시오.

```
# Place import statement outside of function (supported libraries: math, random, numpy,
 scipy, and shapely)
# Example imports of available libraries
```

```
#
# import math
# import random
# import numpy
# import scipy
# import shapely

import math

def reward_function(params):
    ################################################################################
    '''
    Example of using waypoints and heading to make the car point in the right direction
    '''

    # Read input variables
    waypoints = params['waypoints']
    closest_waypoints = params['closest_waypoints']
    heading = params['heading']

    # Initialize the reward with typical value
    reward = 1.0

    # Calculate the direction of the center line based on the closest waypoints
    next_point = waypoints[closest_waypoints[1]]
    prev_point = waypoints[closest_waypoints[0]]

    # Calculate the direction in radius, arctan2(dy, dx), the result is (-pi, pi) in
 radians
    track_direction = math.atan2(next_point[1] - prev_point[1], next_point[0] -
 prev_point[0])
    # Convert to degree
    track_direction = math.degrees(track_direction)

    # Calculate the difference between the track direction and the heading direction of
 the car
    direction_diff = abs(track_direction - heading)
    if direction_diff > 180:
        direction_diff = 360 - direction_diff

    # Penalize the reward if the difference is too large
    DIRECTION_THRESHOLD = 10.0
    if direction_diff > DIRECTION_THRESHOLD:
        reward *= 0.5
```

```
    return float(reward)
```

closest_objects

유형: `[int, int]`

범위: `[(0:len(objects_location)-1), (0:len(objects_location)-1)]`

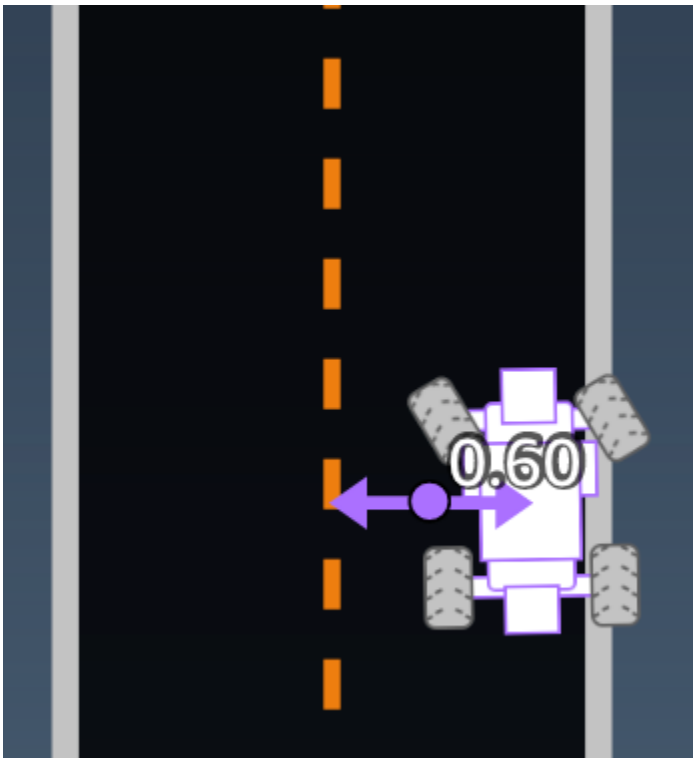
에이전트의 현재 위치(x, y)에 가장 가까운 두 객체의 인덱스(0부터 시작)입니다. 첫 번째 인덱스는 에이전트 뒤에서 가장 가까운 객체를 참조하고 두 번째 인덱스는 에이전트 앞에서 가장 가까운 객체를 참조합니다. 객체가 하나만 있는 경우 두 인덱스는 모두 0이 됩니다.

distance_from_center

유형: `float`

범위: `0:~track_width/2`

에이전트 중앙과 트랙 중앙 사이의 변위(미터)입니다. 에이전트의 바퀴 중 하나라도 트랙 경계를 벗어났을 때 최대 변위가 관측될 수 있으며, 이때 최대 변위는 트랙 경계의 너비에 따라 다르지만 `track_width`의 절반보다 약간 작거나 클 수 있습니다.



예: *distance_from_center* 파라미터를 사용하는 보상 함수

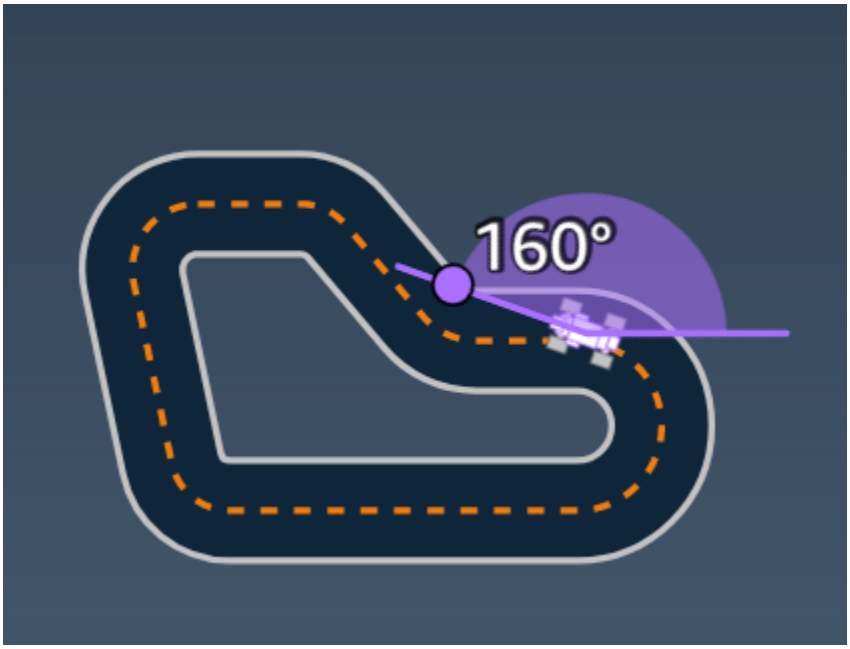
```
def reward_function(params):
    ################################################################################
    '''
    Example of using distance from the center
    '''

    # Read input variable
    track_width = params['track_width']
    distance_from_center = params['distance_from_center']

    # Penalize if the car is too far away from the center
    marker_1 = 0.1 * track_width
    marker_2 = 0.5 * track_width

    if distance_from_center <= marker_1:
        reward = 1.0
    elif distance_from_center <= marker_2:
        reward = 0.5
    else:
        reward = 1e-3  # likely crashed/ close to off track

    return float(reward)
```

heading

유형: `float`

범위: `-180:+180`

좌표계의 x축에 대한 에이전트 진행 방향(각도)입니다.



예: *heading* 파라미터를 사용하는 보상 함수

자세한 내용은 `closest_waypoints` 단원을 참조하십시오.

is_crashed

유형: `Boolean`

범위: `(True:False)`

에이전트가 다른 객체와 충돌했는지(`True`) 또는 충돌하지 않았는지(`False`)를 종료 상태로 나타내는 부울 플래그입니다.

is_left_of_center

유형: `Boolean`

범위: `[True : False]`

에이전트가 트랙 중앙에서 왼쪽에 있는지(`True`), 혹은 오른쪽에 있는지(`False`) 나타내는 `Boolean` 플래그입니다.

is_offtrack

유형: `Boolean`

범위: `(True:False)`

에이전트가 트랙을 벗어났는지(True) 또는 벗어나지 않았는지(False)를 나타내는 부울 플래그입니다.

is_reversed

유형: `Boolean`

범위: `[True:False]`

에이전트가 시계 방향(True) 또는 시계 반대 방향(False)으로 주행하는지 나타내는 부울 플래그입니다.

각 에피소드에 대한 방향 변경을 활성화할 때 사용됩니다.

objects_distance

유형: `[float, … ]`

범위: `[(0:track_length), … ]`

시작선을 기준으로 환경 내 객체 간 거리 목록입니다. i번째 요소는 트랙 중앙선을 따라 i번째 객체와 시작선 사이의 거리(미터)를 측정합니다.

> **Note**
>
> abs | (var1) - (var2)| = how close the car is to an object, WHEN var1 = ["objects_distance"][index] and var2 = params["progress"]*params["track_length"]
> 차량 앞에서 가장 가까운 객체와 차량 뒤에서 가장 가까운 객체의 인덱스를 가져오려면 “closest_object” 파라미터를 사용합니다.

objects_heading

유형: `[float, … ]`

범위: `[(-180:180), … ]`

객체의 방향(도)의 목록입니다. i번째 요소는 i번째 객체의 방향을 측정합니다. 정지 객체의 경우 방향은 0입니다. 로봇 차량의 경우 해당 요소의 값은 차량의 방향 각도입니다.

objects_left_of_center

유형: `[Boolean, … ]`

범위: `[True|False, … ]`

부울 플래그 목록입니다. i번째 요소 값은 i번째 객체가 트랙 중심의 왼쪽(True) 또는 오른쪽(False)에 있는지를 나타냅니다.

objects_location

유형: `[(x,y), … ]`

범위: `[(0:N,0:N), … ]`

모든 객체 위치의 목록으로, 각 위치는 (x, y)의 튜플입니다.

목록 크기는 트랙 위 객체의 수와 같습니다. 객체는 정지 장애물, 이동 로봇 차량이 될 수 있습니다.

objects_speed

유형: `[float, … ]`

범위: `[(0:12.0), … ]`

트랙 위 객체의 속도(m/s) 목록입니다. 정지 객체의 경우 속도는 0입니다. 로봇 차량의 경우 이 값은 훈련 시 설정한 속도입니다.

progress

유형: `float`

범위: `0:100`

주행한 트랙의 비율입니다.

예: *`progress`* 파라미터를 사용하는 보상 함수

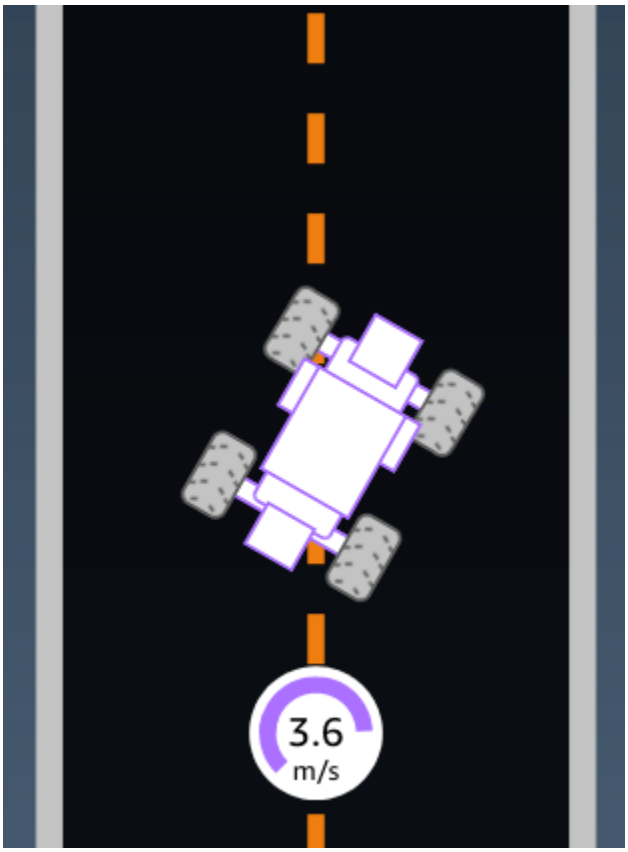
자세한 내용은 단계 항목을 참조하십시오.

속도

유형: `float`

범위: `0.0:5.0`

관측된 에이전트 속도(m/s)입니다.



예: *speed* 파라미터를 사용하는 보상 함수

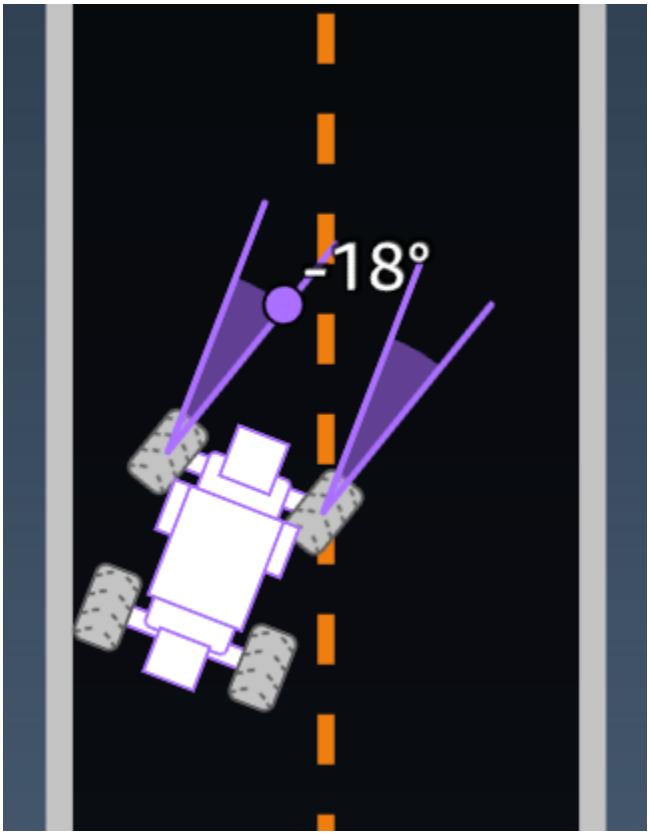
자세한 내용은 all_wheels_on_track 항목을 참조하십시오.

steering_angle

유형: `float`

범위: `-30:30`

에이전트 중앙선에 대한 전륜 조향 각도입니다. 음의 기호(-)는 오른쪽 조향을, 그리고 양의 기호(+)는 왼쪽 조향을 의미합니다. 다음 그림에서 알 수 있듯이 에이전트 중앙선과 트랙 중앙선이 반드시 평행을 이루지는 않습니다.



예: *steering_angle* 파라미터를 사용하는 보상 함수

```
def reward_function(params):
    '''
    Example of using steering angle
    '''

    # Read input variable
    abs_steering = abs(params['steering_angle']) # We don't care whether it is left or
 right steering

    # Initialize the reward with typical value
    reward = 1.0

    # Penalize if car steer too much to prevent zigzag
    ABS_STEERING_THRESHOLD = 20.0
    if abs_steering > ABS_STEERING_THRESHOLD:
        reward *= 0.8

    return float(reward)
```

steps

유형: `int`

범위: $0:N_{step}$

완료한 단계 수입니다. 단계란 현재 정책에 따리 에이전트가 취하는 행동을 말합니다.

예: *steps* 파라미터를 사용하는 보상 함수

```
def reward_function(params):
    ###############################################################################
    '''
    Example of using steps and progress
    '''

    # Read input variable
    steps = params['steps']
    progress = params['progress']

    # Total num of steps we want the car to finish the lap, it will vary depends on the
 track length
    TOTAL_NUM_STEPS = 300

    # Initialize the reward with typical value
    reward = 1.0

    # Give additional reward if the car pass every 100 steps faster than expected
    if (steps % 100) == 0 and progress > (steps / TOTAL_NUM_STEPS) * 100 :
        reward += 10.0

    return float(reward)
```

track_length

유형: `float`

범위: $[0:L_{max}]$

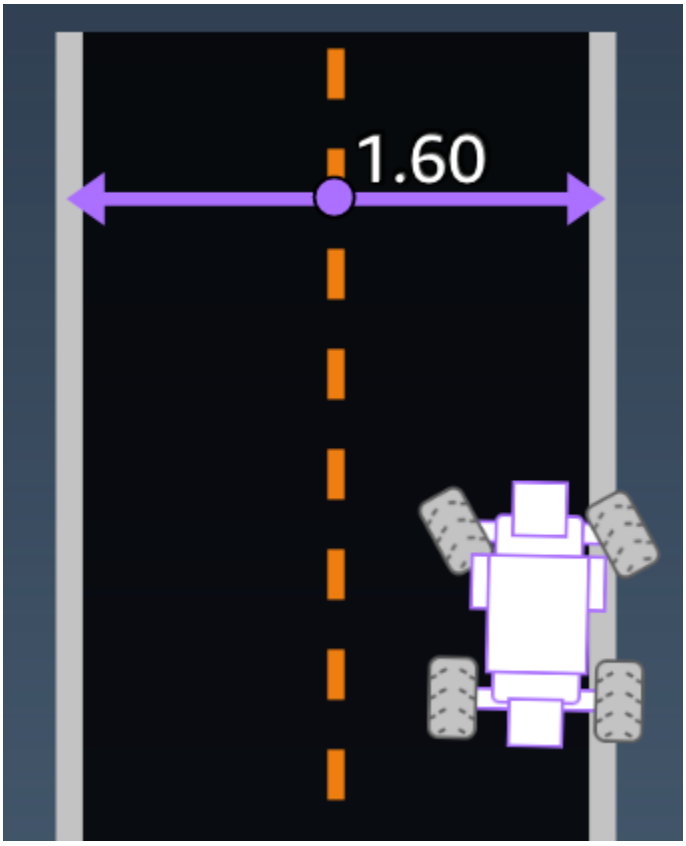
트랙 길이(미터)입니다. L_{max} `is track-dependent.`

track_width

유형: `float`

범위: `0:`D_{track}

트랙 너비(미터)입니다.



예: *`track_width`* 파라미터를 사용하는 보상 함수

```
def reward_function(params):
    ################################################################################
    '''
    Example of using track width
    '''

    # Read input variable
    track_width = params['track_width']
    distance_from_center = params['distance_from_center']

    # Calculate the distance from each border
    distance_from_border = 0.5 * track_width - distance_from_center
```

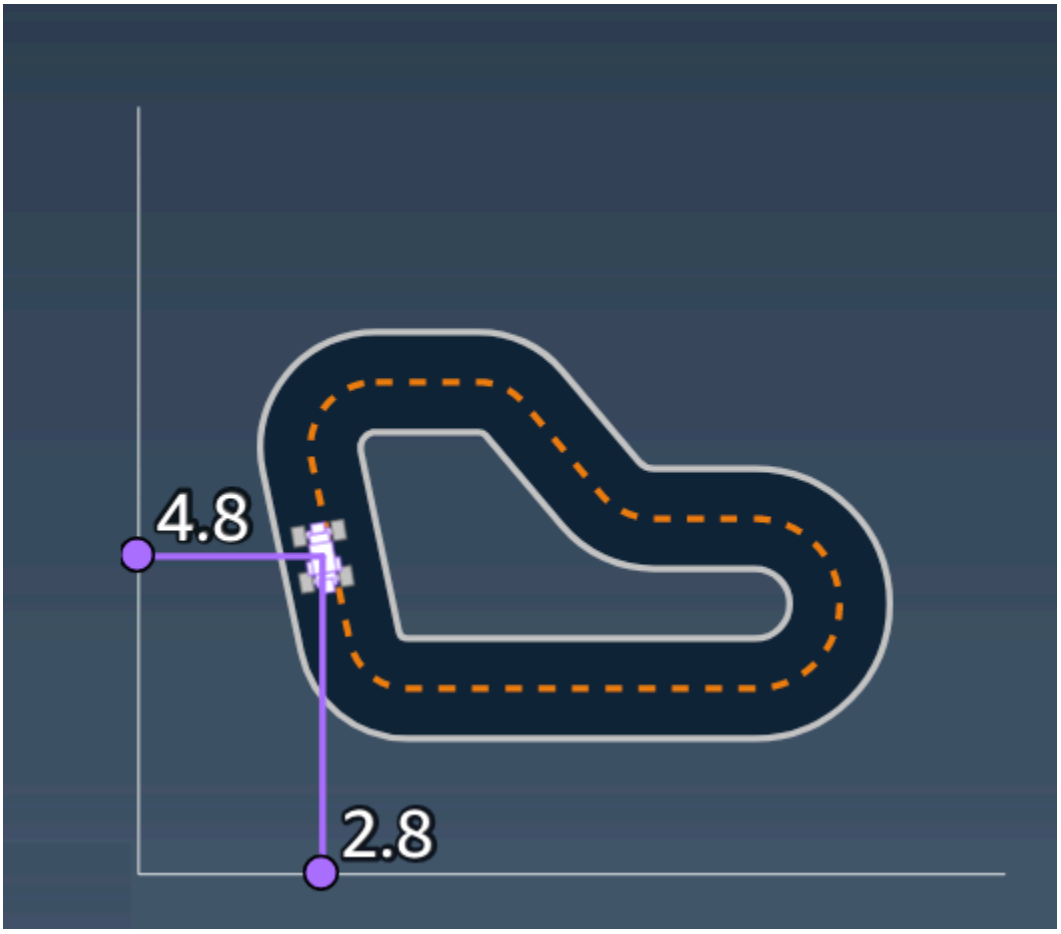
```
    # Reward higher if the car stays inside the track borders
    if distance_from_border >= 0.05:
        reward = 1.0
    else:
        reward = 1e-3 # Low reward if too close to the border or goes off the track

    return float(reward)
```

x, y

유형: `float`

범위: `0:N`

트랙이 포함된 시뮬레이션 환경에서 x축과 y축에 따른 에이전트 중앙의 위치(미터)입니다. 원점은 시뮬레이션 환경에서 왼쪽 하단 모퉁이입니다.

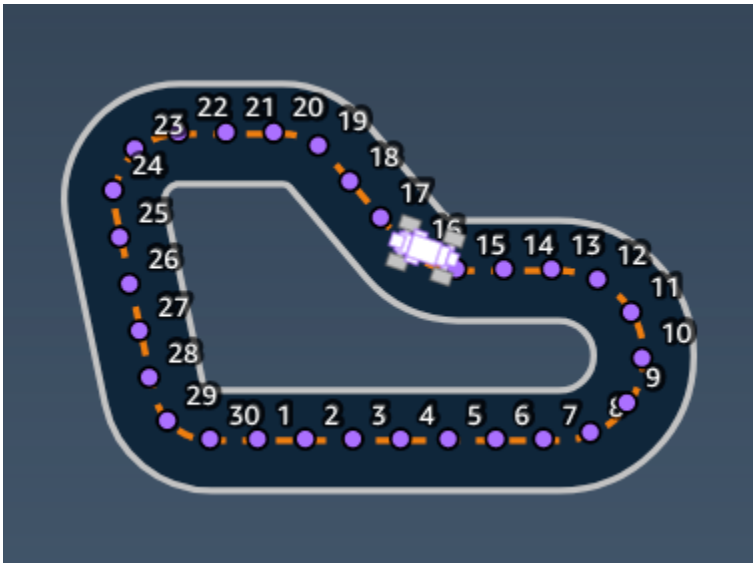


중간 지점

유형: `[float, float] list`

범위: `[[`$x_{w,0}$`,`$y_{w,0}$`] … [`$x_{w,Max-1}$`, `$y_{w,Max-1}$`]]`

트랙 중앙을 따라 순서대로 나열된 트랙에 의존하는 `Max` 이정표 목록입니다. 각 이정표는 ($x_{w,i}$, $y_{w,i}$) 좌표로 알 수 있습니다. 순환 트랙의 경우, 첫 번째와 마지막 중간 지점은 동일합니다. 직선 또는 다른 비순환 트랙의 경우, 첫 번째와 마지막 중간 지점은 다릅니다.



예: *`waypoints`* 파라미터를 사용하는 보상 함수

자세한 내용은 `closest_waypoints` 단원을 참조하십시오.

AWS DeepRacer 보상 함수 예제

다음은 몇 가지 AWS DeepRacer 보상 함수 예제입니다.

주제

- 예제 1: 타임 트라이얼 시 중앙선 따르기
- 예제 2: 타임 트라이얼에서 두 경계선 사이를 유지
- 예제 3: 타임 트라이얼에서 지그재그 주행 방지
- 예제 4: 정지 장애물 또는 이동 차량에 충돌하지 않고 한 차선을 유지

예제 1: 타임 트라이얼 시 중앙선 따르기

이 예제에서는 에이전트가 중앙선에서 얼마나 멀리 떨어져 있는지 확인하고 트랙의 중앙에 가까울 경우 더 높은 보상을 제공하여 에이전트가 중앙선을 가깝게 따르도록 합니다.

```
def reward_function(params):
    '''
    Example of rewarding the agent to follow center line
    '''

    # Read input parameters
    track_width = params['track_width']
    distance_from_center = params['distance_from_center']

    # Calculate 3 markers that are increasingly further away from the center line
    marker_1 = 0.1 * track_width
    marker_2 = 0.25 * track_width
    marker_3 = 0.5 * track_width

    # Give higher reward if the car is closer to center line and vice versa
    if distance_from_center <= marker_1:
        reward = 1
    elif distance_from_center <= marker_2:
        reward = 0.5
    elif distance_from_center <= marker_3:
        reward = 0.1
    else:
        reward = 1e-3  # likely crashed/ close to off track

    return reward
```

예제 2: 타임 트라이얼에서 두 경계선 사이를 유지

이 예제는 에이전트가 경계선 안쪽을 유지하면 높은 보상을 제공하고 에이전트가 랩을 완주하는 데 가장 좋은 경로를 알아내도록 합니다. 프로그래밍하고 이해하기는 쉽지만 수렴하는 데 시간이 오래 걸립니다.

```
def reward_function(params):
    '''
    Example of rewarding the agent to stay inside the two borders of the track
    '''
```

```
    # Read input parameters
    all_wheels_on_track = params['all_wheels_on_track']
    distance_from_center = params['distance_from_center']
    track_width = params['track_width']

    # Give a very low reward by default
    reward = 1e-3

    # Give a high reward if no wheels go off the track and
    # the car is somewhere in between the track borders
    if all_wheels_on_track and (0.5*track_width - distance_from_center) >= 0.05:
        reward = 1.0

    # Always return a float value
    return reward
```

예제 3: 타임 트라이얼에서 지그재그 주행 방지

이 예제는 에이전트가 중앙선을 추종하도록 장려하지만 조향이 너무 많을 경우 보상이 감소하여 지그재그 행동을 방지하는 데 도움이 됩니다. 에이전트가 시뮬레이터에서 원활한 주행 방법을 학습하여 실제 차량에 배치되었을 때도 동일한 행동을 유지할 가능성이 높습니다.

```
def reward_function(params):
    '''
    Example of penalize steering, which helps mitigate zig-zag behaviors
    '''

    # Read input parameters
    distance_from_center = params['distance_from_center']
    track_width = params['track_width']
    abs_steering = abs(params['steering_angle']) # Only need the absolute steering
 angle

    # Calculate 3 marks that are farther and father away from the center line
    marker_1 = 0.1 * track_width
    marker_2 = 0.25 * track_width
    marker_3 = 0.5 * track_width

    # Give higher reward if the car is closer to center line and vice versa
    if distance_from_center <= marker_1:
        reward = 1.0
    elif distance_from_center <= marker_2:
        reward = 0.5
```

```
    elif distance_from_center <= marker_3:
        reward = 0.1
    else:
        reward = 1e-3  # likely crashed/ close to off track

    # Steering penality threshold, change the number based on your action space setting
    ABS_STEERING_THRESHOLD = 15

    # Penalize reward if the car is steering too much
    if abs_steering > ABS_STEERING_THRESHOLD:
        reward *= 0.8

    return float(reward)
```

예제 4: 정지 장애물 또는 이동 차량에 충돌하지 않고 한 차선을 유지

이 보상 함수는 에이전트가 트랙 경계선 사이를 유지하면 보상을 제공하고, 전방 객체에 너무 접근하면 페널티를 부여합니다. 에이전트는 충돌을 회피하기 위해 차선을 변경할 수 있습니다. 총 보상은 보상과 페널티의 가중 합계입니다. 이 예제에서는 충돌을 회피하여 안전에 더 집중하도록 페널티에 더 많은 가중치를 부여합니다. 다양한 평균 가중치를 실험하여 다양한 행동 결과에 대해 훈련합니다.

```
import math
def reward_function(params):
    '''
    Example of rewarding the agent to stay inside two borders
    and penalizing getting too close to the objects in front
    '''
    all_wheels_on_track = params['all_wheels_on_track']
    distance_from_center = params['distance_from_center']
    track_width = params['track_width']
    objects_location = params['objects_location']
    agent_x = params['x']
    agent_y = params['y']
    _, next_object_index = params['closest_objects']
    objects_left_of_center = params['objects_left_of_center']
    is_left_of_center = params['is_left_of_center']
    # Initialize reward with a small number but not zero
    # because zero means off-track or crashed
    reward = 1e-3
    # Reward if the agent stays inside the two borders of the track
    if all_wheels_on_track and (0.5 * track_width - distance_from_center) >= 0.05:
```

```
        reward_lane = 1.0
    else:
        reward_lane = 1e-3
    # Penalize if the agent is too close to the next object
    reward_avoid = 1.0
    # Distance to the next object
    next_object_loc = objects_location[next_object_index]
    distance_closest_object = math.sqrt((agent_x - next_object_loc[0])**2 + (agent_y -
next_object_loc[1])**2)
    # Decide if the agent and the next object is on the same lane
    is_same_lane = objects_left_of_center[next_object_index] == is_left_of_center
    if is_same_lane:
        if 0.5 <= distance_closest_object < 0.8:
            reward_avoid *= 0.5
        elif 0.3 <= distance_closest_object < 0.5:
            reward_avoid *= 0.2
        elif distance_closest_object < 0.3:
            reward_avoid = 1e-3  # Likely crashed
    # Calculate reward by putting different weights on
    # the two aspects above
    reward += 1.0 * reward_lane + 4.0 * reward_avoid
    return reward
```

AWS DeepRacer 콘솔에서 모델 가져오기 및 내보내기

AWS DeepRacer 모델을 가져오거나 내보내야 하는 시나리오가 있습니다. 고용주가 후원하는 이벤트에 참가한 레이서는 모델을 내보내 액세스 권한을 잃지 않도록 할 수 있으며, 레이스 관리자는 참석자가 이벤트 중에 가져와 사용할 수 있도록 사전 훈련된 모델을 제공할 수 있습니다. 사용자 모델 페이지를 사용하여 콘솔에서 AWS DeepRacer 모델을 가져오고 내보낼 수 있습니다.

주제

- AWS DeepRacer 모델을 Amazon S3에 복사
- AWS DeepRacer 모델을 콘솔로 가져오기
- 문제 해결

AWS DeepRacer 모델을 Amazon S3에 복사

AWS DeepRacer 모델을 Amazon S3에 복사하는 방법

1. AWS DeepRacer 콘솔에 로그인합니다.
2. 탐색 창의 강화 학습에서 모델을 선택합니다.
3. 모델 옆에 있는 확인란을 선택하여 가져오려는 모델을 선택합니다. 콘솔에서 한 번에 하나의 모델만 Amazon S3로 복사할 수 있습니다.
4. 작업 버튼 드롭다운을 선택한 다음 S3로 복사를 선택합니다.

   Amazon S3로 복사 페이지가 새로 열립니다.
5. Amazon S3로 복사 페이지에서 Amazon S3 버킷 드롭다운 선택기를 사용하여 모델을 내보낼 Amazon S3 버킷을 선택합니다. AWS DeepRacer S3 버킷은 이름에 딥페이서를 포함해야 합니다.

   - 유효한 Amazon S3 버킷이 없는 경우 새 버킷 생성을 선택하여 하나 생성합니다. 드롭다운 선택기는 버킷 이름을 형식 `aws-deepracer-assets-XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX`으로 채웁니다.
6. S3 객체 접두사 필드의 Amazon S3 객체에 선택적 폴더 접두사를 추가합니다.
7. S3 버킷을 설정한 후 포함하려는 자산을 선택합니다. 하나 이상의 자산 유형을 선택해야 합니다.

   - 모델: 모델 폴더에는 모델 가져오기에 필요한 모든 파일이 들어 있습니다.

- 로그: 모델의 훈련 및 평가 로그를 복사합니다. 이 옵션에는 `logs/`, `metrics/`, 및 `sim-trace/` 폴더가 포함됩니다.
- 비디오: 이 옵션은 비디오 폴더를 Amazon S3 버킷에 복사합니다. 비디오 폴더에는 `evaluation/` 및 `training/` 폴더가 들어 있습니다. 이 폴더에는 평면도, 45도 각도 보기, 45도 각도 보기가 포함되며 콘솔 오버레이에는 트랙 위의 차량 위치가 표시됩니다.

8. 복사를 누르면 Amazon S3 데이터 스토리지 비용을 부담해야 한다는 팝업이 나타납니다. 약관에 동의하면 팝업에서 복사 버튼을 누릅니다.
9. 복사 프로세스가 시작되면 콘솔의 내 모델 페이지로 돌아갑니다. 페이지 상단의 배너에 현재 상태가 표시됩니다. 내보내기 프로세스가 완료되면 배너에서 내보내기 성공 여부를 확인할 수 있습니다.

모델 가져오기에 필요한 파일

콘솔 외부에서 훈련된 모델의 모델 폴더를 업로드하려면 Amazon S3 설명서의 객체 업로드 페이지에 있는 단계를 따릅니다. 다음 표에는 모델 가져오기에 필요한 파일 목록이 포함되어 있습니다. 필수 파일이 하나라도 없으면 모델 가져오기가 실패합니다.

AWS DeepRacer 콘솔에서 훈련된 모델은 폴더 이름 형식 `DAY/MONTH/YEAR/TIME GMT`을 갖습니다. 예제 모델은 2023년 11월 30일에 내보냈으며 폴더 이름은 `Thu, 30 Nov 2023 19:01:24 GMT`입니다. 이 예제에서는 이 폴더를 루트라고 합니다.

모델 가져오기에 필요한 파일

파일 이름	폴더 경로	설명
.coach_checkpoint	root/model/	코치 체크포인트 파일에는 가져오기에 사용된 모델 체크포인트의 키가 들어 있습니다.
ckpt 파일	root/model/	체크포인트 파일은 훈련 중 여러 단계에서 촬영한 모델 가중치의 스냅샷입니다. `ckpt.index` , `ckpt.data` 및 `ckpt.meta` 파일이 포함됩니다.
model_metadata.json	root/	모델 메타데이터 파일에는 작업 공간 정의, 센서 구성, 훈련

파일 이름	폴더 경로	설명
		알고리즘 선택 등이 포함된 설정이 포함되어 있습니다.
reward_function.py	root/	모델 훈련에 사용되는 보상 함수가 들어 있는 Python 파일.

지표 파일은 모델을 가져오는 데 필요하지 않습니다. 이러한 파일이 포함되지 않은 경우, 해당 모델의 훈련 지표와 보상 그래프를 콘솔에서 사용할 수 없습니다.

모델 가져오기를 위한 옵션 파일

파일 이름	폴더 경로	설명
training_params.yaml	root/	training_params 파일에는 트랙 및 차량 정보, 레이서 및 모델 이름, 훈련 아티팩트의 폴더 경로 등이 포함된 훈련 작업 데이터가 들어 있습니다.
hyperparameters.json	root/ip/	배치 크기, 손실 유형, 학습률, 에포크 수와 같은 모델의 하이퍼파라미터 정보를 포함합니다.
training-*.json	root/metrics/training/	AWS DeepRacer 콘솔에서 모델의 훈련 지표를 시각화하는 데 사용됩니다.

AWS DeepRacer 모델을 콘솔로 가져오기

이 섹션에서는 AWS DeepRacer 모델을 콘솔로 가져오는 절차를 단계별로 살펴봅니다. 모델을 가져오려면 먼저 모델 폴더의 Amazon S3 URL을 복사해야 합니다.

AWS DeepRacer Amazon S3 bucket URL 복사

1. Amazon S3 콘솔에 로그인하고 버킷 페이지로 이동합니다.

2. 버킷 이름에 있는 링크를 눌러 AWS DeepRacer에 생성한 Amazon S3 버킷을 선택합니다. AWS DeepRacer 콘솔에서 생성되는 S3 버킷의 형식은 `aws-deepracer-assets-XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX`입니다.
3. AWS DeepRacer 버킷의 객체 탭에서:
   a. 이름 필드에서 모델 객체의 이름 링크를 눌러 가져오려는 모델을 선택합니다.

      하위 폴더 목록이 나타납니다.
   b. 폴더 이름 옆의 확인란을 선택하여 모델 폴더의 루트를 선택합니다. AWS DeepRacer 콘솔에서 훈련된 모델은 폴더 이름 형식 `DAY/MONTH/YEAR/TIME GMT`을 갖습니다.
4. 모델 폴더의 루트로 이동한 후 S3 URL 복사 버튼을 선택합니다. 모델 이름 옆의 확인란을 선택할 때까지 s3 URL 복사, URL 복사, 열기 및 삭제 버튼이 회색으로 표시됩니다.

> **Note**
>
> S3로 복사하는 과정 중에 *`my_model/version_2`*와 같은 접두사를 추가한 경우 모델 폴더의 경로는 `deep_racer_bucket/model_name/`*`my_model/version_2`*`/root/`입니다.

모델을 AWS DeepRacer 콘솔로 가져오기

1. AWS DeepRacer 콘솔에서 사용자 모델 페이지로 이동합니다.
2. 모델 컨테이너에서 모델 가져오기 버튼을 선택합니다.

   모델 가져오기 페이지가 나타납니다.
3. 가져오기 섹션에서:
   - 가져오려는 모델 폴더의 Amazon S3 URL을 입력합니다. Amazon S3 URL의 형식은 `s3://`*`deep_racer_bucket/model_name/prefix/root`*입니다.
4. 세부 정보 섹션에서:
   a. 모델 이름을 입력합니다.
   b. 선택 사항으로 모델에 대한 설명을 추가합니다.
   c. 다중 사용자 모드를 사용하는 관리자 계정을 사용하는 경우 드롭다운 선택기에서 모델을 가져오려는 사용자를 선택합니다.
5. 화면 아래쪽에 있는 가져오기 버튼을 선택합니다.

6. 가져오기 프로세스가 시작되면 콘솔의 사용자 모델 페이지로 돌아갑니다. 페이지 상단의 배너에는 현재 상태가 표시되고 모델은 가져오는 중... 과 함께 모델 목록에 상태 그대로 나타납니다. 가져오기 프로세스가 완료되면 배너에서 가져오기 성공 여부를 확인하고 모델 상태가 가져오는 중...에서 준비 완료로 변경됩니다.

문제 해결

모델 복사 오류

여러 번 시도했지만 모델을 복사할 수 없습니다. 모델이 아직 S3 버킷에 있는 경우 모델 오류 테이블에서 모델을 선택하고 업데이트를 선택한 다음 가져오기를 선택하여 모델 가져오기를 다시 시도합니다. 또는 모델의 로컬 사본이 있는 경우 Amazon S3 설명서의 객체 업로드 페이지에 있는 단계에 따라 모델을 수동으로 가져올 수 있습니다.

Amazon S3 버킷이 존재하지 않음

이 모델이 저장된 S3 버킷이 삭제되었으므로 모델을 복사할 수 없습니다. 모델 사본이 있는 경우 이름의 `deepracer`를 포함한 S3 버킷에 모델을 배치하고 AWS DeepRacer 모델을 콘솔로 가져오기 섹션의 단계에 따라 다시 가져옵니다.

Amazon S3 버킷에 액세스할 수 없음

이 모델이 저장된 Amazon S3 버킷의 권한이 변경되어 모델을 복사할 수 없습니다. 이는 두 가지 이유 때문에 발생할 수 있습니다. 즉, AWS DeepRacer S3 또는 AWS DeepRacer 서비스 역할 정책에서 권한을 직접 편집했기 때문입니다. AWS DeepRacer S3 버킷에서 권한을 직접 편집한 경우 Amazon S3 콘솔 페이지를 사용하여 버킷 정책 추가 페이지의 단계에 따라 다음 정책을 사용하여 버킷 권한을 복원합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1586917903457",
            "Effect": "Allow",
            "Principal": {
                "Service": "deepracer.amazonaws.com"
                },
            "Action": [
                "s3:GetObjectAcl",
```

```
                "s3:GetObject",
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": [
               "arn:aws:s3:::your-bucket-name",
               "arn:aws:s3:::your-bucket-name/*"
            ]
        }
    ]
}
```

버킷 권한을 복원한 후 모델을 가져오려면 모델 오류 테이블에서 모델을 선택하고 업데이트를 선택합니다. 모델 가져오기 페이지가 나타나면 가져오기를 선택합니다.

모델 파일이 존재하지 않음

Amazon S3 버킷에서 모델이 삭제되었으므로 모델을 복사할 수 없습니다. 파일이 아직 있는 경우 파일을 AWS DeepRacer 버킷으로 복원한 다음 모델 오류 테이블에서 모델을 선택하고 업데이트를 선택합니다. 모델 가져오기 페이지가 나타나면 가져오기를 선택합니다. 모델의 로컬 사본이 있는 경우 Amazon S3 설명서의 객체 업로드 페이지에 있는 단계에 따라 파일을 수동으로 가져올 수 있습니다.

코치 파일이 존재하지 않음

코치 체크포인트 메타데이터가 Amazon S3 버킷에서 삭제되었으므로 모델을 복사할 수 없습니다. 파일이 아직 있는 경우 파일을 AWS DeepRacer 버킷으로 복원한 다음 모델 오류 테이블에서 모델을 선택하고 업데이트를 선택합니다. 모델 가져오기 페이지가 나타나면 가져오기를 선택합니다. 모델의 로컬 사본이 있는 경우 Amazon S3 설명서의 객체 업로드 페이지에 있는 단계에 따라 파일을 수동으로 가져올 수 있습니다.

체크포인트 파일이 존재하지 않음

Amazon S3 버킷에서 체크포인트 파일이 삭제되었으므로 모델을 복사할 수 없습니다. 여전히 파일이 있는 경우 파일을 AWS DeepRacer 버킷으로 복원하고 모델 오류 테이블에서 모델을 선택한 다음 업데이트를 선택합니다. 모델 가져오기 페이지가 나타나면 가져오기를 선택합니다. 파일의 로컬 사본이 있는 경우 Amazon S3 설명서의 객체 업로드 페이지에 있는 단계에 따라 파일을 수동으로 가져올 수 있습니다.

모델 파일이 너무 큽니다

모델 파일이 서비스에서 생성할 수 있는 1GB 파일 크기 제한을 초과하여 파일이 편집되었습니다. 이 모델은 가져올 수 없습니다. 이 메시지를 제거하려면 모델 오류 테이블에서 모델을 선택한 다음 삭제를 선택합니다.

체크포인트 파일이 너무 큽니다

체크포인트 파일이 서비스에서 생성할 수 있는 파일 크기 한도 1GB를 초과하여 파일이 편집되었습니다. 이 모델은 가져올 수 없습니다. 이 메시지를 제거하려면 모델 오류 테이블에서 모델을 선택한 다음 삭제를 선택합니다.

메타데이터 파일이 너무 큽니다

YAML 파일이 서비스에서 생성할 수 있는 10MB 파일 크기 제한을 초과하여 파일이 편집되었습니다. 이 모델은 가져올 수 없습니다. 이 메시지를 제거하려면 모델 오류 테이블에서 모델을 선택한 다음 삭제를 선택합니다.

모델이 유효하지 않음

모델이 편집되었기 때문에 모델을 검증할 수 없습니다. 모델 사본이 있는 경우, AWS DeepRacer S3 버킷에서 해당 모델을 교체해 본 후 모델 오류 테이블에서 모델을 선택한 다음 업데이트를 선택합니다. 모델 가져오기 페이지가 나타나면 가져오기를 선택합니다.

권한이 없거나 올바르지 않음

훈련 당시 AWS DeepRacer에서 사용할 수 있었던 권한이 제거되었기 때문에 모델을 복사할 수 없었습니다. AWS DeepRacer가 필요한 권한을 다시 생성하도록 승인하려면 모델 오류 테이블에서 모델을 선택한 다음 업데이트를 선택합니다. 모델 가져오기 페이지가 나타나면 가져오기를 선택합니다. AWS DeepRacer는 권한을 다시 생성한 다음 모델을 복사합니다.

AWS DeepRacer 차량 작동

AWS DeepRacer 시뮬레이터에서 AWS DeepRacer 모델 훈련과 평가를 마쳤으면 이제 훈련된 모델을 AWS DeepRacer 차량에 배포할 수 있습니다. 트랙에서 주행할 수 있도록 차량을 설정한 후 물리적 환경에서 모델 성능을 평가하면 됩니다. 방법은 실제 자율 레이스와 똑같습니다.

차량을 처음 주행할 때는 먼저 차량을 설정하고, 소프트웨어 업데이트를 설치한 다음 드라이브 체인 하위 계통을 보정해야 합니다.

또한 물리적 트랙에서 차량을 주행하려면 트랙이 필요합니다. 자세한 정보는 물리적 트랙 빌드 섹션을 참조하십시오.

주제

- AWS DeepRacer 차량 알아보기
- AWS DeepRacer 차량에 사용할 Wi-Fi 네트워크 선택
- AWS DeepRacer 차량의 디바이스 콘솔을 시작합니다.
- AWS DeepRacer 차량을 보정하십시오.
- AWS DeepRacer 차량에 모델 업로드
- AWS DeepRacer 차량을 운전해 보십시오.
- AWS DeepRacer 차량 설정 검사 및 관리
- AWS DeepRacer 차량 로그 보기

AWS DeepRacer 차량 알아보기

이 AWS DeepRacer 차량은 기계 학습이 가능한 1/18 크기의 4륜 구동 모형 자동차로서 Wi-Fi를 지원하고, 배터리 전력 공급 방식이고, 4메가픽셀 전방 카메라와 Ubuntu 기반 컴퓨팅 모듈이 탑재되어 있습니다.

이 차량은 컴퓨팅 모듈의 강화 학습 모델에 따라 추론을 실행하여 자율 주행할 수 있습니다. 또한 강화 학습 모델을 배포하지 않고 수동으로도 주행 가능합니다. 아직 AWS DeepRacer 차량이 없다면 여기에서 주문할 수 있습니다.

AWS DeepRacer 차량은 브러시 모터로 움직입니다. 주행 속도는 모터 회전 속도를 제어하는 전압 조정기를 통해 제어됩니다. 조향 디바이스를 구동하는 서보 메커니즘(서보)은 AWS DeepRacer 차량 섀시에서 검은색 커버로 보호됩니다.

주제

- AWS DeepRacer 차량을 검사하십시오.
- AWS DeepRacer 배터리 충전 및 설치
- AWS DeepRacer 컴퓨팅 모듈을 테스트하십시오.
- AWS DeepRacer 차량의 전원을 끕니다.
- AWS DeepRacer 차량 LED 표시등
- AWS DeepRacer 디바이스 예비 부품

AWS DeepRacer 차량을 검사하십시오.

AWS DeepRacer 차량 상자를 개봉하면 다음과 같은 구성품과 액세서리가 있어야 합니다.

Components	설명
차량 섀시[1]	차량 주행 경험을 수집하기 위한 전방 카메라와 자율 주행을 위한 컴퓨팅 모듈이 내장됩니다. 카메라에서 수집되는 이미지는 차량의 디바이스 콘

Components	설명
	솔에서 스트리밍 동영상으로 확인할 수 있습니다. 그 밖에도 섀시에는 브러시 전기 모터와 전자식 속도 컨트롤러(ESC), 그리고 서보 메커니즘(서보)가 내장됩니다.
차체[2]	차량을 설정할 때 이 구성품을 분리합니다.
Micro USB-USB A 케이블 [3]	USB-OTG 기능을 지원할 때 사용됩니다.
컴퓨팅 배터리 [4]	다운로드한 AWS DeepRacer 강화 학습 모델에 대해 추론을 실행하는 컴퓨팅 모듈에 전력을 공급할 때 사용됩니다.
컴퓨팅 배터리 커넥터 케이블 [5]	컴퓨팅 모듈을 배터리와 연결할 때 이 USB C-USB C 케이블이 사용됩니다. Dell 컴퓨팅 배터리를 사용하는 경우 이 케이블은 더 깁니다.
전원 케이블 [6a]	전원 어댑터를 전원 콘센트에 연결할 때 사용됩니다.
전원 어댑터 [6b]	컴퓨팅 배터리와 컴퓨팅 모듈을 충전할 때 사용됩니다.
핀(예비 부품) [7]	컴퓨팅 모듈을 차량 섀시에 고정할 때 사용됩니다. 이 부품은 추가적인 부품입니다.
차량 배터리 [8]	모터에 전력을 공급하는 7.4v LiPo 배터리 팩입니다.
차량 배터리 충전 어댑터 [9a]	차량 드라이브 체인에 전력을 공급하는 차량 배터리를 충전할 때 사용됩니다.
차량 배터리 충전 케이블 [9b]	차량 배터리 충전기를 전원 콘센트에 연결할 때 사용됩니다.
배터리 잠금 해제 케이블 [10]	배터리가 잠금 상태로 전환되면 이를 사용합니다.

AWS DeepRacer 차량을 설정하려면 먼저 다음 항목이 준비되어 있어야 합니다.

- USB 포트가 장착되고, 인터넷 액세스가 가능한 컴퓨터
- 인터넷에 연결된 Wi-Fi 네트워크
- AWS 계정.

이제 다음 단원 지침에 따라 차량 배터리와 파워 뱅크가 충전되어 있는지 확인합니다.

AWS DeepRacer 배터리 충전 및 설치

AWS DeepRacer 차량에는 차량 배터리와 컴퓨팅 모듈 파워 뱅크라는 두 개의 전원 공급 디바이스가 있습니다.

파워 뱅크는 컴퓨팅 모듈이 계속해서 실행되도록 전력을 공급합니다. 컴퓨팅 모듈은 Wi-Fi 연결을 유지하고, 배포된 AWS DeepRacer 모델에 대해 추론을 실행하고, 차량이 행동할 수 있도록 명령을 실행합니다.

차량 배터리는 모터에 전력을 공급하여 차량을 구동합니다. 두 세트의 케이블이 있습니다. 빨간색 및 검은색 케이블의 두 개의 연결된 세트는 차량의 ESC에 연결하는데 사용되며 세 개의 연결된 파란색(또는 검은색), 흰색, 빨간색 케이블은 충전기를 연결하는데 사용됩니다. 주행할 때는 두 개의 연결된 케이블 세트만 차량에 연결되어야 합니다.

배터리는 완전히 충전된 이후 방전되면서 전압도 떨어집니다. 전압이 떨어지면 유효 토크도 낮아집니다. 그 결과, 동일한 속도 설정에서도 트랙에서 더 느린 속도가 될 수 있습니다. 배터리가 완전히 방전되면 차량이 움직이지 않습니다. 정상적인 조건에서 자율 주행할 경우 배터리의 지속 시간은 15~25분입니다. 일관적인 동작을 유지하려면 사용 후 15분마다 배터리를 충전하는 것이 좋습니다.

차량 배터리와 파워 뱅크를 설치하고 충전하는 단계는 아래와 같습니다.

1. AWS DeepRacer 차량 쉘을 제거하십시오.
2. 차량 섀시 핀 4개를 분리합니다. 와이어가 연결된 상태에서 차량 섀시를 주의하여 들어 올립니다.
3. 다음과 같은 방법으로 차량 배터리를 충전하고 설치합니다.

   a. 배터리를 충전하려면 배터리의 세 개의 연결된 케이블 세트를 충전기에 꽂아 배터리를 전원 어댑터에 연결하고 전원 어댑터를 벽 콘센트 또는 USB 케이블을 사용하여 배터리를 충전하는 경우, USB 포트에 꽂습니다.

      동봉된 충전기를 사용하여 차량 배터리를 충전하는 방법에 대한 그래픽 설명은 the section called “차량의 주행 모듈 배터리를 충전하는 방법”를 참조하십시오.

b. 배터리가 충전되면, 차량 배터리 케이블의 두 개의 연결된 케이블 세트를 차량의 검은색/빨간색 케이블 커넥터에 연결합니다.

c. 배터리 차량을 고정하려면 첨부된 스트랩을 사용해 배터리를 차량 섀시 밑에 묶습니다.

   이때 모든 케이블이 차량 내부로 들어가야 합니다.

d. 다음과 같은 방법으로 차량 배터리의 충전 여부를 확인합니다.

   i. 차량 전원 스위치를 밀어서 차량을 켭니다.

   ii. 짧은 신호음이 2회 들리는지 확인합니다.

      신호음이 들리지 않으면 차량이 충전되지 않은 상태입니다. 차량에서 배터리를 제거하고 위의 1단계를 반복하여 배터리를 재충전합니다.

   iii. 차량을 사용하지 않을 때, 차량 전원 스위치를 원위치로 밀어서 차량 배터리를 끕니다.

4. 다음과 같은 방법으로 파워 뱅크의 충전 레벨을 확인합니다.

   a. 파워 뱅크의 전원 버튼을 누릅니다.

   b. 전원 버튼 옆에 있는 LED 조명 4개를 사용해 충전 레벨을 확인합니다.

      LED 조명 4개가 모두 켜지면 파워 뱅크가 완전히 충전된 상태입니다. LED 조명이 하나도 켜지지 않으면 파워 뱅크를 충전해야 합니다.

   c. 파워 뱅크를 충전하려면 전원 어댑터의 USB C 플러그를 파워 뱅크의 USB C 포트에 삽입합니다. 파워 뱅크가 완전히 충전되려면 시간이 약간 걸립니다. 충전되면 Step 4(4단계)를 반복하여 파워 뱅크가 완전히 충전되었는지 확인합니다.

5. 다음과 같은 방법으로 파워 뱅크를 설치합니다.

   a. 전원 버튼과 USB C 포트가 차량 후면을 향하는 상태에서 파워 뱅크를 홀더에 삽입합니다.

   b. 스트랩을 사용해 파워 뱅크를 차량 섀시에 단단히 묶습니다.

> **Note**
>
> 이번 단계에서는 파워 뱅크를 컴퓨팅 모듈에 연결하지 마십시오.

AWS DeepRacer 컴퓨팅 모듈을 테스트하십시오.

컴퓨팅 모듈을 테스트하여 정상적으로 시작되는지 확인합니다. 외부 전원 공급 디바이스를 사용해 다음과 같은 방법으로 모듈을 테스트합니다.

차량의 컴퓨팅 모듈을 테스트하는 방법

1. 컴퓨팅 모듈을 전원 공급 디바이스에 연결합니다. 전원 코드를 전원 어댑터와 전원 콘센트에 연결한 후 전원 어댑터의 USB C 플러그를 컴퓨팅 모듈의 USB C 포트에 삽입합니다.
2. 컴퓨팅 모듈의 전원 버튼을 눌러 차량의 컴퓨팅 모듈을 켭니다.
3. 다음과 같이 LED 조명을 점검하여 컴퓨팅 모듈의 상태를 확인합니다.
   - 파란색

     컴퓨팅 모듈이 시작되었고, 지정된 Wi-Fi에 연결되어 있고, 사용할 준비가 되었습니다.

     이 상태에서는 HDMI 케이블, USB 마우스 및 USB 키보드를 사용해 모니터에 연결한 후 컴퓨팅 모듈에 로그인할 수 있습니다. 처음 로그인할 경우에는 `deepracer`를 사용자 이름과 암호로 사용합니다. 그러면 이후 로그인할 때는 암호를 재설정하라는 메시지가 표시됩니다. 새로운 암호로 재설정할 때는 보안을 이유로 강력한 암호 문구를 선택해야 합니다.
   - 점멸 빨간색

     컴퓨팅 모듈이 설정 모드입니다.
   - 노란색

     컴퓨팅 모듈이 초기화 중입니다.
   - 빨간색

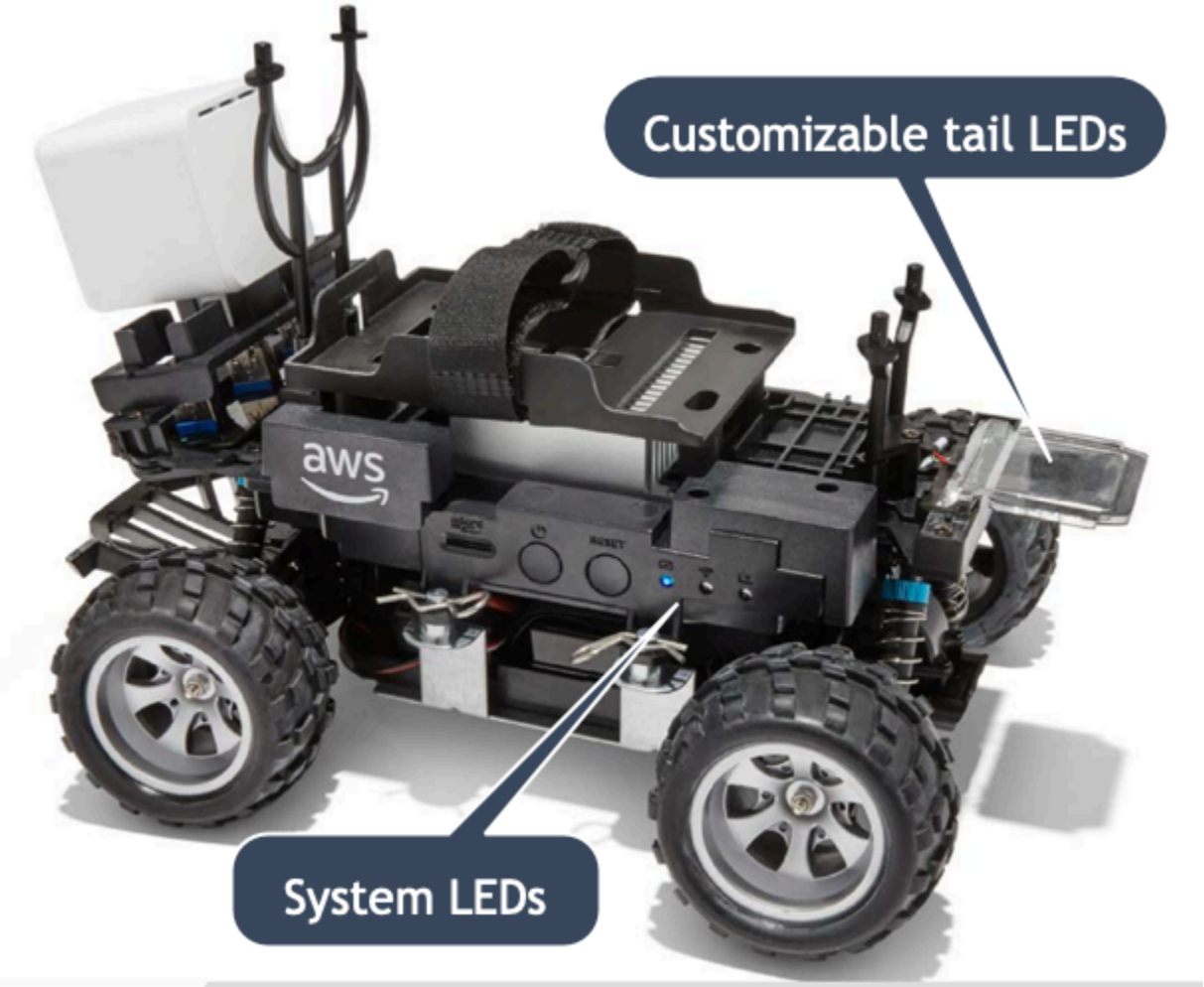
     컴퓨팅 모듈이 Wi-Fi 네트워크에 연결되지 않았습니다.
4. 테스트를 마치면 컴퓨팅 모듈의 전원 버튼을 눌러 끈 다음 컴퓨팅 모듈을 외부 전원 공급 디바이스에서 뽑습니다.

AWS DeepRacer 차량의 전원을 끕니다.

AWS DeepRacer 차량 전원을 끌 때는 차량을 외부 전원 공급 디바이스에서 뽑습니다. 또한 전원 표시등이 꺼질 때까지 디바이스의 전원 버튼을 누르는 방법도 있습니다.

AWS DeepRacer 차량 LED 표시등

AWS DeepRacer 차량에는 두 세트의 LED 표시등이 있습니다. 하나는 차량 상태를 나타내고 다른 하나는 사용자 지정 가능한 시각적 차량 식별에 사용됩니다.



세부 정보는 아래에서 다룹니다.

주제

- AWS DeepRacer 차량 시스템 LED 표시등
- AWS DeepRacer 차량 식별 LED

AWS DeepRacer 차량 시스템 LED 표시등

차량이 전방 위치에 있을 때 AWS DeepRacer 차량 시스템 LED 표시등은 차량 섀시의 왼쪽에 위치합니다.

3개의 시스템 LED가 재설정 버튼 뒤에 위치합니다. 첫 번째 LED(시야 왼쪽)는 시스템 전원 상태를 보여줍니다. 두 번째(가운데) LED는 향후 사용을 위해 예약됩니다. 마지막(오른쪽) LED는 Wi-Fi 연결 상태를 보여줍니다.

LED 유형	색상	상태 표시기
Power	꺼짐	전원 공급 장치가 없습니다.
	노란색 깜박임	BIOS 및 OS가 로드 중입니다.
	노란색 켜짐	OS가 로드되었습니다.
	파란색 켜짐	애플리케이션이 실행 중입니다.
	파란색 깜박임	소프트웨어 업데이트가 진행 중입니다.
	빨간색 켜짐	시스템이 부팅 중이거나 애플리케이션이 시작 중일 때 오류가 발생했습니다.
Wi-Fi	꺼짐	Wi-Fi 연결이 없습니다.
	파란색 깜박임	차량이 Wi-Fi 네트워크에 연결 중입니다.
	2초 동안 빨간색 켜진 후 꺼짐	Wi-Fi 연결에 실패했습니다.
	파란색 켜짐	Wi-Fi 연결이 설정되었습니다.

AWS DeepRacer 차량 식별 LED

AWS DeepRacer 차량 사용자 지정 LED는 차량의 테일에 위치합니다. 이는 여러 차량이 존재할 때 주행 중인 차량을 식별하는 데 사용됩니다. AWS DeepRacer 디바이스 콘솔을 사용하여 원하는 지원 색상으로 설정할 수 있습니다.

AWS DeepRacer 디바이스 예비 부품

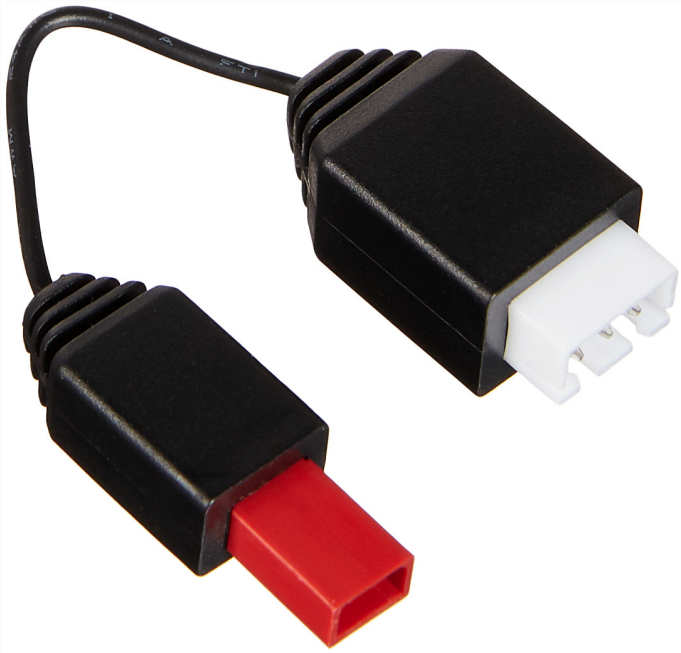
> **Note**
>
> AWS DeepRacer 디바이스는 WLToys A949 및 A979 원격 제어(RC) 자동차 섀시를 사용합니다. AWS DeepRacer 디바이스에서 사용 가능한 최신 부품 목록을 찾아보려면 AWS DeepRacer 스토어를 방문하십시오.

예비 AWS DeepRacer 디바이스 부품

부품	명칭
	예비 컴퓨팅 배터리

부품	명칭
	예비 컴퓨팅 배터리

부품	명칭
	예비 컴퓨팅 배터리

부품	명칭
	리튬 배터리 7.4V 1100mAh
	AWS DeepRacer 자동차 배터리 잠금 해제 케이블
	타이어
	앞 범퍼

부품	명칭
	서스펜션 암
	풀 로드
	C 스타일 시트
	변속기 샤프트
	둥근 머리 나사, M2x17.5mm
	섀시 자동차 하단
	회전 시트

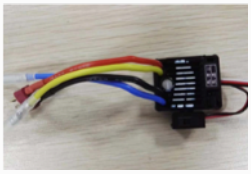
부품	명칭
	리어 서스펜션 프레임
	금속 육각 컴바이너 세트
	기어 박스 쉘
	디퍼렌셜 박스 케이스
	디퍼렌셜 드라이브 컵
	프런트 리어 쉘터
	서보 시트

부품	명칭
	중앙 드라이빙 샤프트
	쇼크 프레임
	서보 암
	디퍼렌셜 메커니즘
	감속 기어
	모터 베이스
	17g 스티어링 엔진

부품	명칭
	모터의 나사 개스킷, 고정 시트
	390 모터
	육각 커넥터 4x8x3mm
	육각 커넥터 8x12x3.5mm
	볼 베어링 7x11x3mm
	볼 베어링 8x12x3.5mm
	중간 액슬 디스크 플레이트

부품	명칭
	나사 2.6x6mm
	나사 2x7mm
	나사 2.5x8mm
	나사 2x16mm
	나사 2.5x6x6mm
	나사 M3x5mm
	볼 나사 10.8x4mm

부품	명칭
	나사 2x6mm
	나사 2x9.5mm
	M3 로크 너트
	액슬 힌지 핀
	드라이브 샤프트
	스윙 암 핀
	나사 2*29KM

부품	명칭
	헤어 핀
	프런트 쇼크 업소버
	충전기
	금속 모터 피니언 기어
	리어 쇼크 업소버
	ESC

AWS DeepRacer 차량에 사용할 Wi-Fi 네트워크 선택

AWS DeepRacer 차량을 처음 열 경우 Wi-Fi 네트워크 연결을 설정해야 합니다. 차량의 소프트웨어를 업데이트하거나 IP 주소를 가져와 차량의 디바이스 콘솔에 액세스하려면 이 설정을 마쳐야 합니다.

이번 단원에서는 다음 작업의 실행 단계에 대해서 살펴보겠습니다.

- 노트북 또는 데스크톱 컴퓨터를 차량에 연결합니다.
- 차량의 Wi-Fi 연결을 설정합니다.
- 차량의 소프트웨어를 업데이트합니다.
- 차량의 IP 주소를 가져옵니다.
- 차량을 테스트 드라이브합니다.

노트북 또는 데스크톱 컴퓨터를 사용해 설정 작업을 실행합니다. 지금부터는 Ubuntu 운영 체제 기반 차량 컴퓨팅 모듈과 혼동을 피하기 위해 이 설정 컴퓨터를 컴퓨터라고 부르겠습니다.

처음 Wi-Fi 연결 설정을 마친 뒤 동일한 지침에 따라 다른 Wi-Fi 네트워크도 선택할 수 있습니다.

> **Note**
>
> AWS DeepRacer에서는 로그인할 때 활성 캡차 확인을 요구하는 Wi-Fi 네트워크를 지원하지 않습니다.

주제

- AWS DeepRacer 차량의 Wi-Fi 연결 설정 준비
- Wi-Fi 연결 설정 및 AWS DeepRacer 차량의 소프트웨어 업데이트

AWS DeepRacer 차량의 Wi-Fi 연결 설정 준비

차량의 Wi-Fi 연결을 설정하려면 함께 제공되는 USB-to-USB C 케이블을 사용해 노트북 또는 데스크톱 컴퓨터를 차량의 컴퓨팅 모듈에 연결해야 합니다.

컴퓨터를 차량의 컴퓨팅 모듈에 연결하는 단계는 아래와 같습니다.

1. 디바이스를 연결하기 전에 컴퓨터가 Wi-Fi 연결을 끊었는지 확인하십시오.
2. USB-to-USB C 케이블의 USB 단부를 컴퓨터의 USB 포트에 삽입합니다.
3. 케이블의 USB C 단부를 차량의 USB C 포트에 삽입합니다.

이제 차량의 Wi-Fi 연결을 설정할 준비를 마쳤습니다.

Wi-Fi 연결 설정 및 AWS DeepRacer 차량의 소프트웨어 업데이트

이번 단원 단계에 따라 Wi-Fi 연결을 설정하려면 먼저 the section called “ Wi-Fi 설정 준비 ” 단원의 단계를 완료해야 합니다.

1. 차량 하단을 보고 호스트 이름 아래 인쇄된 암호를 기록합니다. 이 암호는 설정을 위해 디바이스 제어 콘솔에 로그인할 때 필요합니다.
2. 컴퓨터에서 `https://deepracer.aws`로 이동하여 차량의 디바이스 제어 콘솔을 시작합니다.
3. 연결이 비공개가 아니거나 안전하지 않다는 메시지가 표시되면 다음 중 한 가지를 실행합니다.
   a. Chrome을 사용하는 경우에는 고급을 선택한 다음 ***<device_console_ip_address>***로 진행(안전하지 않음)을 선택합니다.
   b. Safari를 사용하는 경우에는 세부 정보를 선택한 후 이 웹사이트 방문 링크를 따라 웹사이트 방문를 선택합니다. 인증서 신뢰 설정을 업데이트하려면 암호를 입력하라는 메시지가 표시되면 암호를 입력하고 설정 업데이트를 선택합니다.
   c. Opera를 사용하는 경우에는 잘못된 인증서라는 경고 메시지가 표시되어도 계속을 선택합니다.
   d. Edge를 사용하는 경우에는 세부 정보와 웹사이트로 이동(권장 안 함)를 차례대로 선택합니다.
   e. Firefox를 사용하는 경우에는 고급, 예외 추가, 보안 예외 확인을 차례대로 선택합니다.
4. Under AWS DeepRacer 차량 잠금 해제 아래 1단계에서 기록한 암호를 입력하고 차량 액세스를 선택합니다.
5. 차량을 Wi-Fi 네트워크에 연결 창에 있는 Wi-Fi 네트워크 이름(SSID) 드롭다운 메뉴에서 원하는 Wi-Fi 네트워크 이름을 선택하고 Wi-Fi 암호 아래 Wi-Fi 네트워크 암호를 입력한 후 연결을 선택합니다.
6. Wi-Fi 연결 상태가 Wi-Fi 네트워크에 연결 중...에서 연결 상태로 바뀔 때까지 기다립니다. 그리고 다음을 선택합니다.
7. 소프트웨어 업데이트창에서 소프트웨어 업데이트가 필요하면 함께 제공되는 전원 코드와 어댑터를 사용해 차량의 컴퓨팅 모듈을 켜고 소프트웨어 업데이트 설치를 선택합니다.

   외부 전력 공급 장치를 사용해 차량 전력을 공급하면 컴퓨팅 모듈의 파워 뱅크가 방전되더라도 소프트웨어 업데이트의 간섭을 방지하는 데 효과적입니다.
8. 소프트웨어 업데이트 상태가 소프트웨어 업데이트 설치 중에서 소프트웨어 업데이트가 성공적으로 설치됨로 바뀔 때까지 기다립니다.

9. Wi-Fi 네트워크 세부 정보 아래 표시되는 IP 주소를 기록합니다. 초기 설정 및 나중에 Wi-Fi 네트워크 설정을 수정한 후 차량의 디바이스 제어 콘솔을 열려면 이 IP 주소가 필요합니다.

AWS DeepRacer 차량의 디바이스 콘솔을 시작합니다.

차량의 Wi-Fi 연결을 설정하고 필요한 소프트웨어 업데이트를 설치했다면 이제 디바이스 콘솔을 열어 차량의 네트워크 연결이 유효한지 확인해야 합니다. 그런 다음 디바이스 콘솔을 시작하여 기타 차량 설정을 검사하고, 보정하고, 관리할 수 있습니다. 이 과정에서 차량 IP 주소를 사용하여 차량의 디바이스 콘솔에 로그인해야 합니다.

디바이스 제어 콘솔은 차량에서 호스팅하며, Wi-Fi 설정 단원 끝에서 가져온 IP 주소를 사용해 액세스합니다.

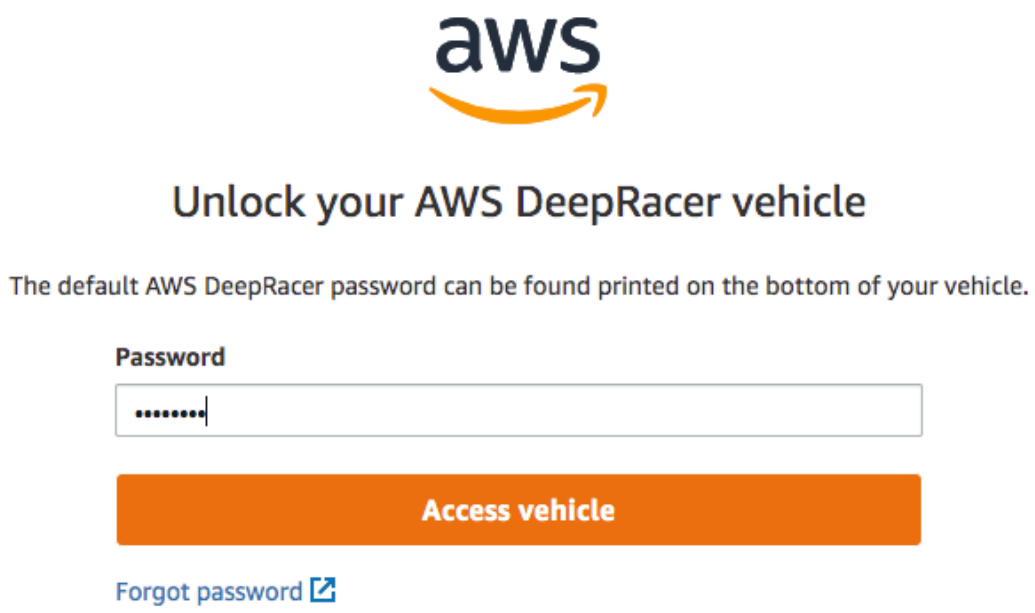
Wi-Fi 연결을 통해 AWS DeepRacer 차량의 디바이스 콘솔에 액세스하는 방법

1. 차량의 디바이스 콘솔에 액세스하려면 먼저 컴퓨터, 태블릿 또는 스마트폰에서 웹 브라우저를 열고 차량의 IP 주소를 주소 창에 입력합니다.

   이 IP 주소는 차량의 Wi-Fi 연결을 설정하면서 가져올 수 있습니다. 설명을 위해 `10.92.206.61`을 예로 들어 사용하겠습니다.

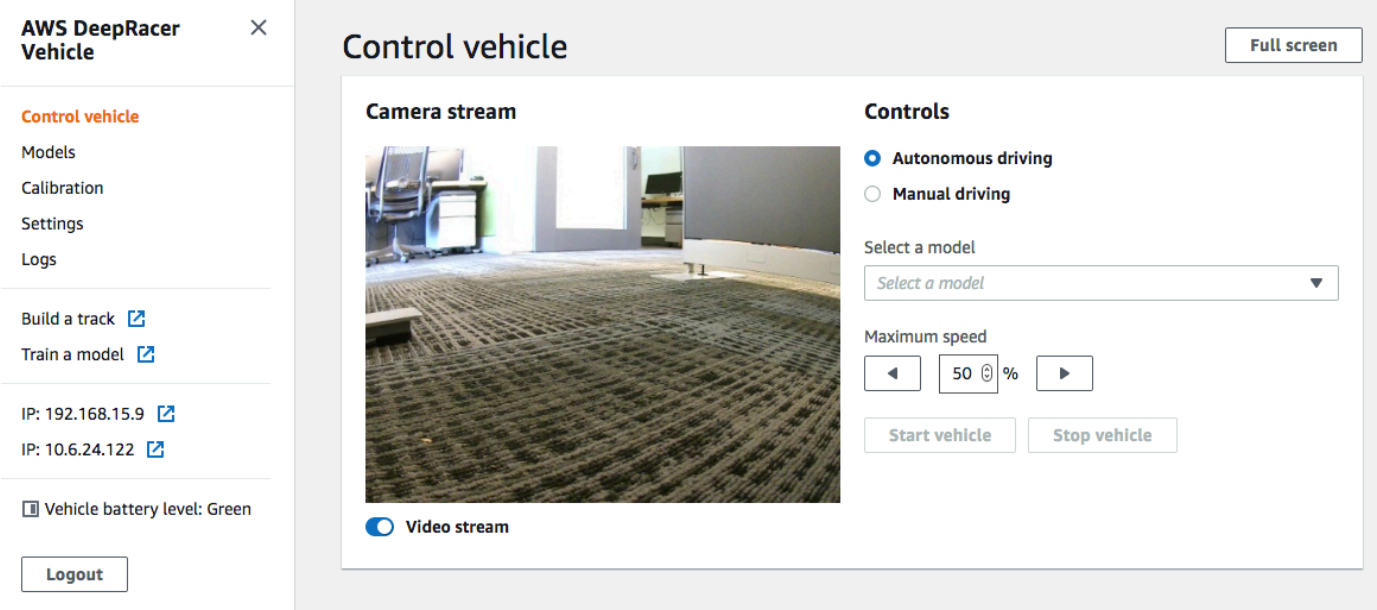
   연결이 비공개가 아니거나 안전하지 않다는 경고 메시지가 표시되더라도 메시지를 무시하고 디바이스 콘솔에 연결합니다.

2. AWS DeepRacer 차량 잠금 해제 아래에서 디바이스 콘솔의 암호를 암호에 입력한 후 차량 액세스을 선택합니다.



기본 암호는 차량 아래쪽(호스트 이름 아래)에 인쇄되어 있습니다.

3. 로그인하면 디바이스 콘솔의 홈 페이지가 다음과 같이 표시됩니다.



이제 차량을 보정하고 작동할 수 있습니다. 차량을 처음 구동하는 경우에는 차량 보정 단원으로 진행하십시오.

AWS DeepRacer 차량을 보정하십시오.

최적의 성능을 얻기 위해서는 AWS DeepRacer 차량의 물리적 부품 몇 가지를 보정해야 합니다. 보정되지 않은 차량을 사용하면 모델을 테스트할 때 불확실성이 커질 수 있습니다. 또한 차량의 성능이 최적화되지 않은 경우에는 딥 너링 모델 코드만 조정하게 될 수도 있습니다. 하지만 근본 원인이 기계적이라면 코드 조정으로는 차량 성능을 개선할 수 없습니다. 이때는 보정을 통해 기계를 조정해야 합니다.

AWS DeepRacer 차량을 보정하려면 먼저 차량의 전자 제어 시스템(ECS)과 서보 메커니즘(서보)에 맞게 듀티 사이클 범위를 각각 설정합니다. 서보와 ECS 모두 펄스폭 변조(PWM)) 신호를 차량 컴퓨팅 모듈의 제어 입력값으로 수신합니다. 컴퓨팅 모듈은 PWM 신호의 듀티 사이클을 변경하여 차량의 속도와 조향 각도를 조정합니다.

최대 속도 및 조향 각도에 따라 행동 공간의 범위가 결정됩니다. 최대 속도와 최대 조향 각도는 시뮬레이션 훈련에서 지정할 수 있습니다. 실제 트랙에서 주행할 목적으로 훈련된 모델을 AWS DeepRacer 차량에 배포할 때는 차량의 최대 속도 및 조향 각도를 시뮬레이션 훈련에서 사용하는 최대 속도 및 조향 각도와 일치하도록 보정해야 합니다.

실제 경험이 시뮬레이션 경험과 일치하도록 하려면 시뮬레이션과 실제 사이에서 최대 속도 및 최대 조향 각도가 일치하도록 차량을 보정해야 합니다. 일반적으로 이러한 보정 방법은 두 가지입니다.

- 훈련 행동 공간을 정의한 후 설정이 일치하도록 물리적 차량을 보정합니다.
- 차량의 실제 성능을 측정한 후 시뮬레이션에서 훈련 공간 설정을 변경합니다.

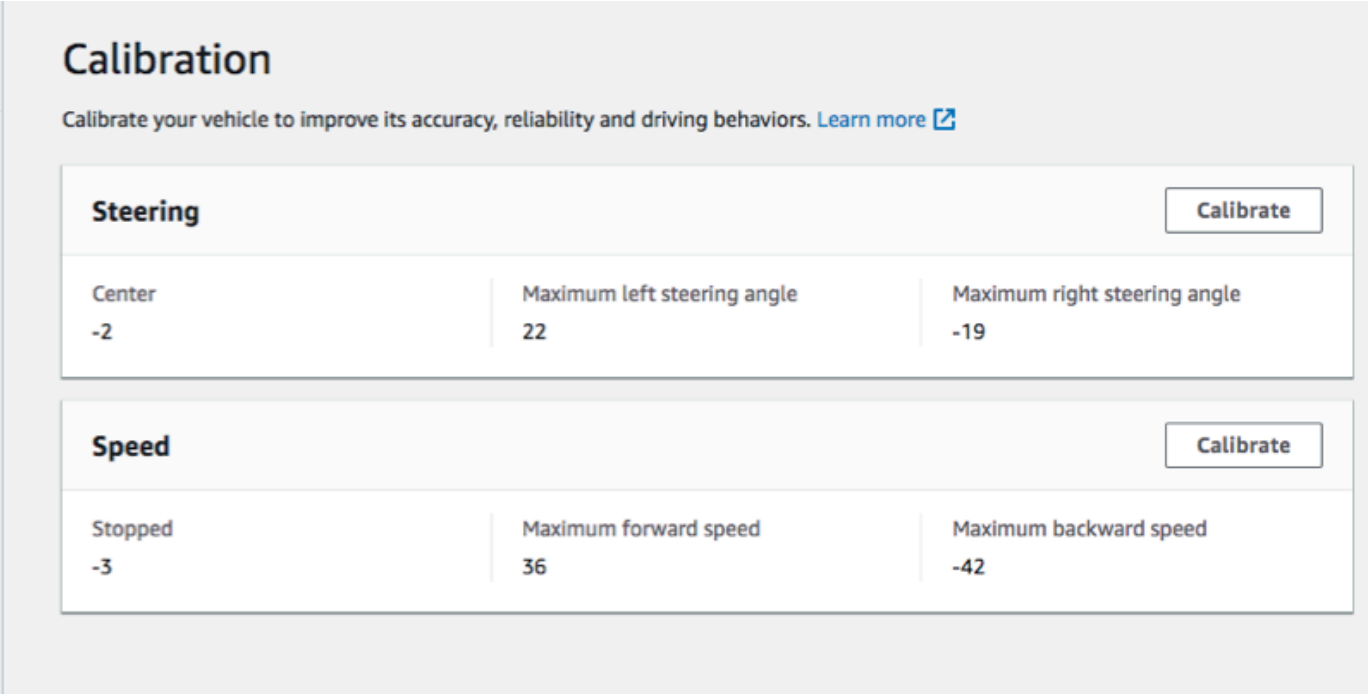
견고한 모델이라면 시뮬레이션과 실제 사이에서 발생하는 차이까지 어느 정도 처리할 수 있습니다. 하지만 두 방식 중 하나로 실험하면서 최상의 결과를 얻을 때까지 반복해야 합니다.

보정을 시작하려면 먼저 컴퓨팅 모듈을 켭니다. 컴퓨팅 모듈이 시작되면 전원 LED가 파란색으로 바뀌면서 차량 배터리가 작동합니다. 짧은 신호음 2회와 긴 신호음 1회가 들리면 보정을 시작할 수 있습니다.

훈련 설정과 일치하도록 AWS DeepRacer 차량을 보정하는 방법:

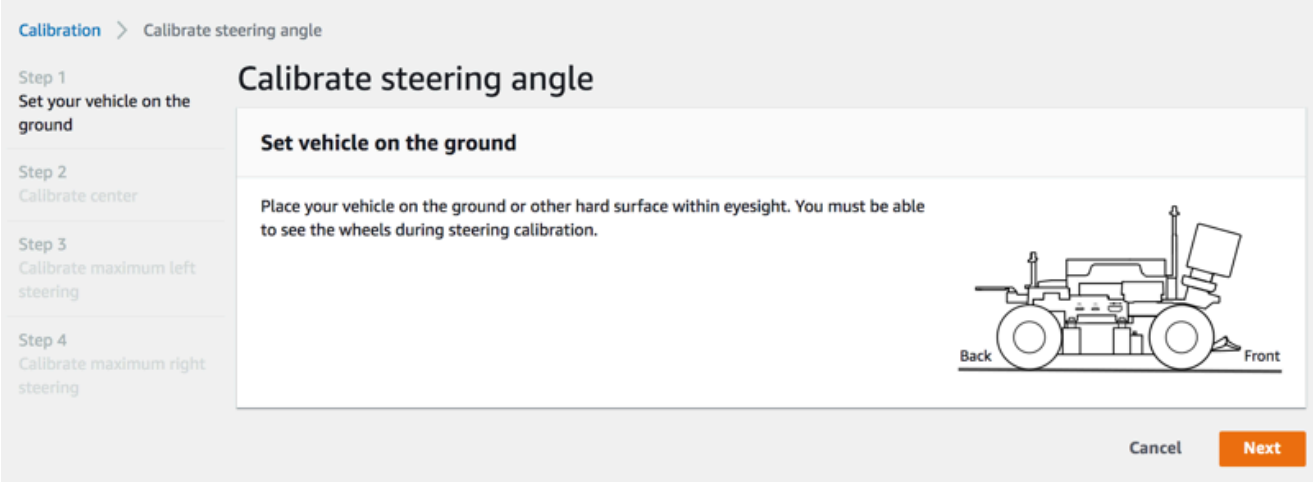
1. 이 지침에 따라 차량에 액세스하여 장치 제어 콘솔을 엽니다.

2. 기본 탐색 창에서 보정을 선택합니다.



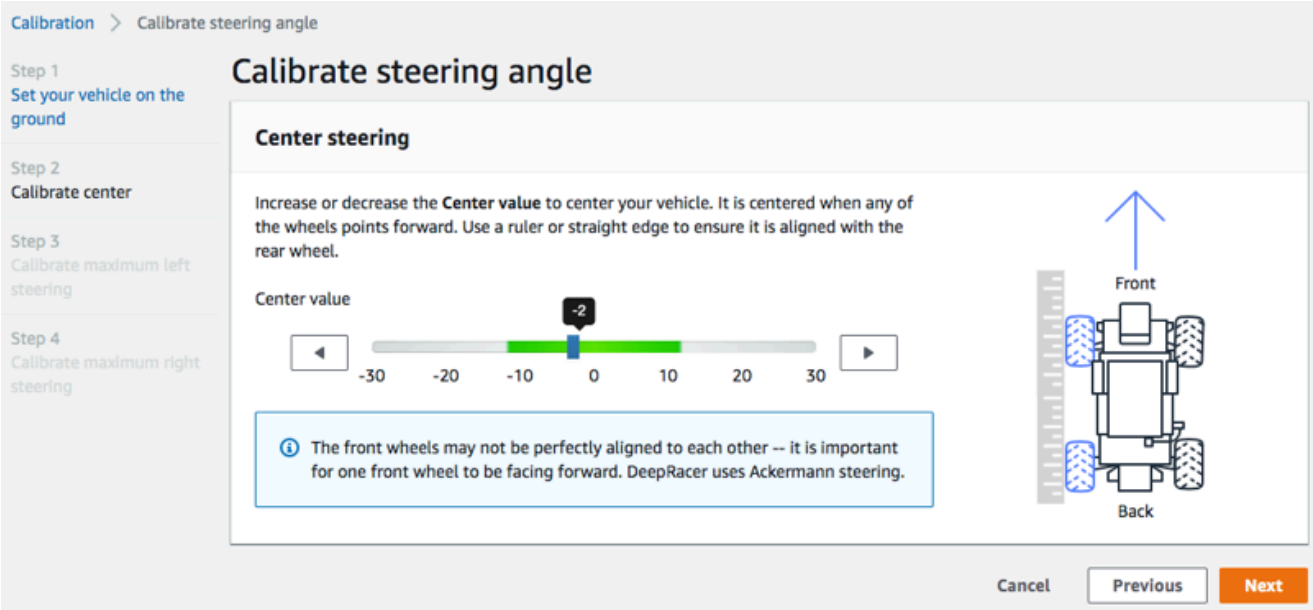
3. 보정 페이지에서 조향 보정을 선택한 다음 아래 단계에 따라 차량의 최대 조향 각도를 보정하십시오.

   a. 조향을 보정할 때는 바퀴를 볼 수 있는 바닥 또는 기타 단단한 표면에 차량을 내려 놓으십시오. Next(다음)를 선택합니다.



   트랙에서 차량을 조향하려면 공중의 바퀴 회전보다 훨씬 작은 조향 각도가 필요합니다. 따라서 실제 바퀴의 조향 각도를 측정하려면 차량을 트랙 표면에 내려놓아야 합니다.

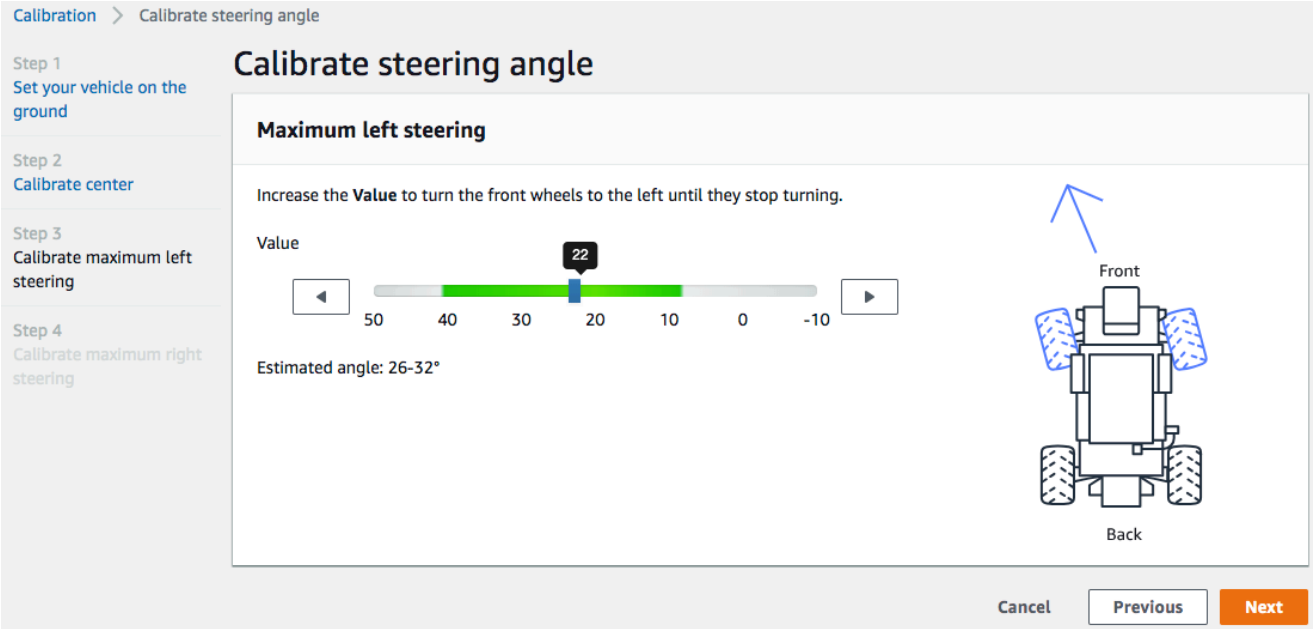
   b. 중앙 조향 각도 아래에서 슬라이더를 움직이거나, 혹은 왼쪽/오른쪽 화살표를 눌러 전륜 하나가 동일한 측면의 후륜과 정렬되는 위치로 조정합니다. Next(다음)를 선택합니다.



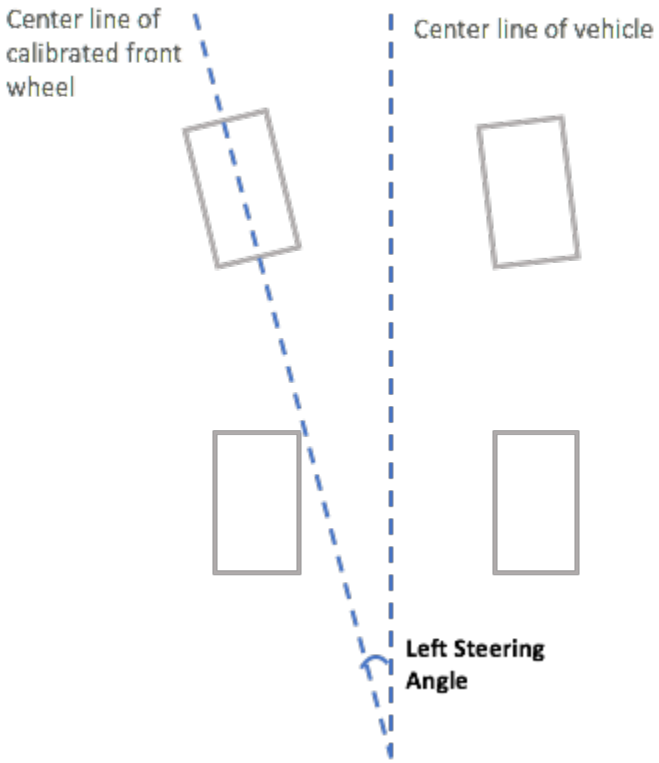
AWS DeepRacer는 애커먼 전륜 조향을 사용해 회전 안쪽과 바깥쪽의 바퀴를 회전시킵니다. 이 말은 일반적으로 왼쪽/오른쪽 전륜의 회전 각도가 서로 다르다는 것을 의미합니다. AWS DeepRacer에서는 중앙 값을 기준으로 보정이 이루어집니다. 따라서 선택한 측면의 바퀴가 직선으로 정렬되도록 조정해야 합니다.

> **Note**
>
> AWS DeepRacer 차량을 보정하여 중앙 조향 각도가 최대한 직선을 유지하도록 해야 합니다. 직선 여부는 차량을 손으로 밀었을 때 직선 경로를 따르는지 테스트하여 알아낼 수 있습니다.

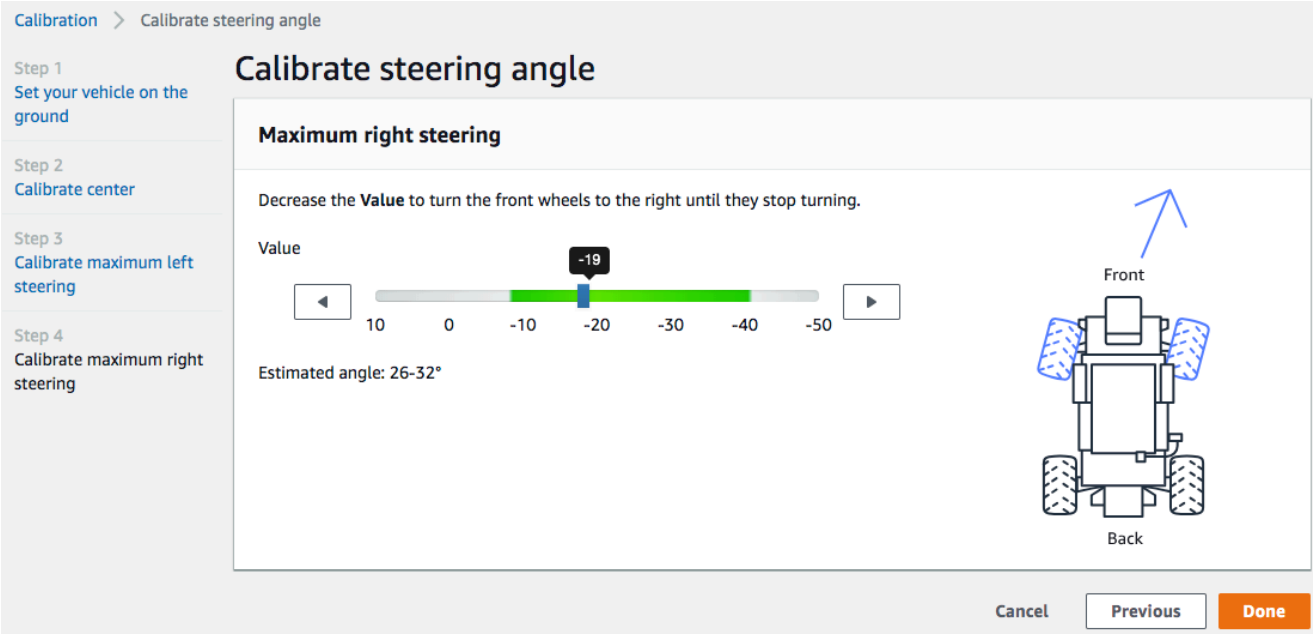
c. 최대 왼쪽 조향 각도 아래에서 차량 전륜이 왼쪽 회전을 멈출 때까지 슬라이더를 왼쪽으로 조금씩 움직이거나 왼쪽 화살표를 누릅니다. 이때 작은 잡음이 일어납니다. 큰 잡음이 들리면 지나치게 움직인 것입니다. 이 위치가 최대 왼쪽 조향 각도입니다. 시뮬레이션 행동 공간에서 조향 각도가 제한적인 경우에는 여기에서 해당 값과 일치하게 조정합니다. Next(다음)를 선택합니다.



실제 최대 왼쪽 조향 각도를 측정하려면 차량 중앙선을 그리고 보정할 전륜의 두 모서리 점을 표시한 다음 차량 중앙선과 교차할 때까지 전륜 중앙선을 그립니다. 각도기를 사용해 각도를 측정합니다. 아래 그림을 참조하십시오. 실제 훈련 각도를 일치시키려면 다음 훈련 작업에서 행동 공간 값을 동일하게 설정하면 됩니다.



d. 최대 오른쪽 조향 각도 아래에서 선택한 전륜이 오른쪽 회전을 멈출 때까지 슬라이더를 오른쪽으로 조금씩 움직입니다. 이때 작은 잡음이 일어납니다. 큰 잡음이 들리면 지나치게 움직인 것입니다. 이 위치가 최대 오른쪽 조향 각도입니다. 시뮬레이션 행동 공간에서 조향 각도가 제한적인 경우에는 여기에서 해당 값과 일치하게 조정합니다. 완료를 선택합니다.

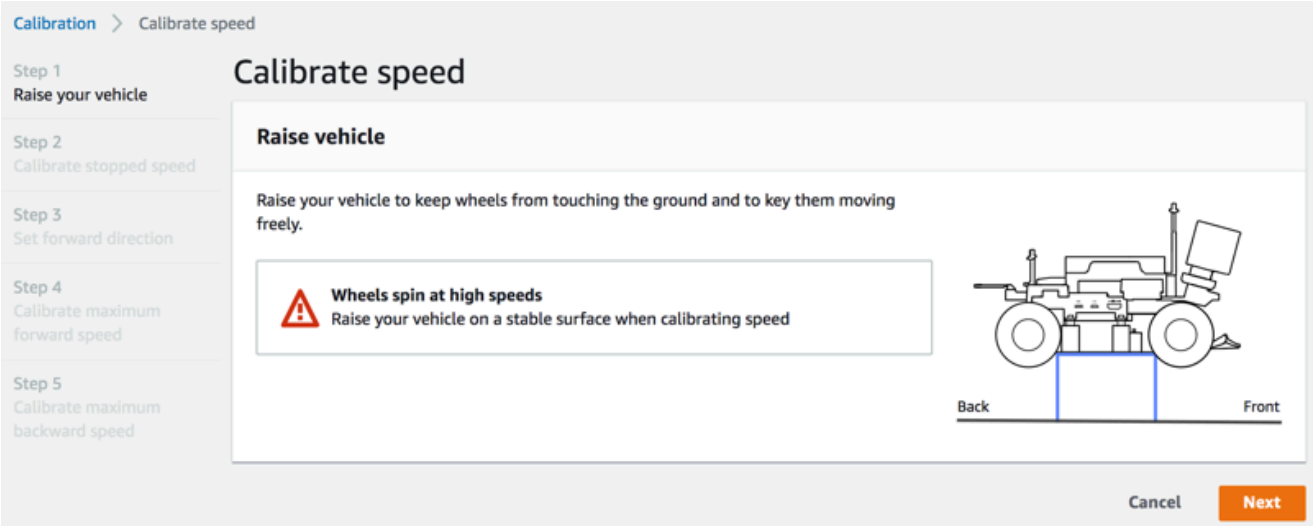


실제 최대 오른쪽 조향 각도를 측정하려면 최대 왼쪽 조향 각도를 측정할 때 사용한 방법과 비슷하게 각 단계를 따릅니다.

이것으로 AWS DeepRacer 차량의 조향 각도 보정이 완료됩니다.

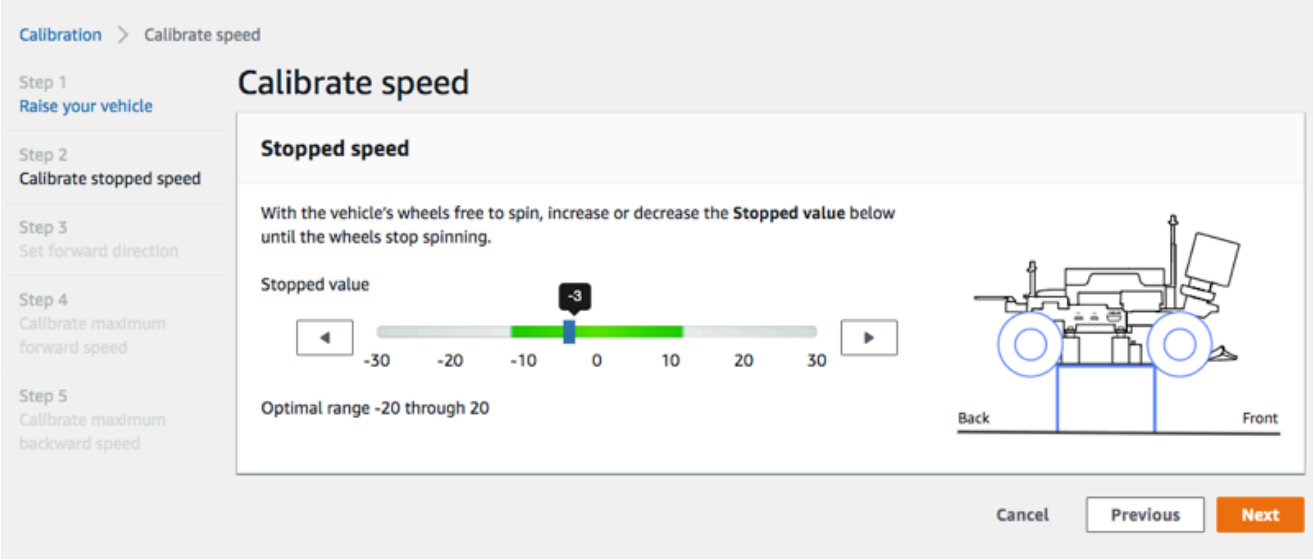
4. 차량의 최대 속도를 보정하려면 보정 페이지에서 속도 보정을 선택한 다음 아래 단계를 따르십시오.

a. 바퀴가 자유롭게 회전할 수 있도록 차량을 들어 올립니다. 디바이스 제어 콘솔에서 다음을 선택합니다.



> **Note**
>
> 차량의 속도가 너무 높게 설정되어 있으면 보정 과정에서 너무 빠르게 작동하여 자칫 환경, 차량 또는 기타 주변 물체가 손상될 수 있습니다. 여기에서 설명하는 대로 차량을 들어 올려야 하지만 양손으로 붙잡아서도 안 됩니다.

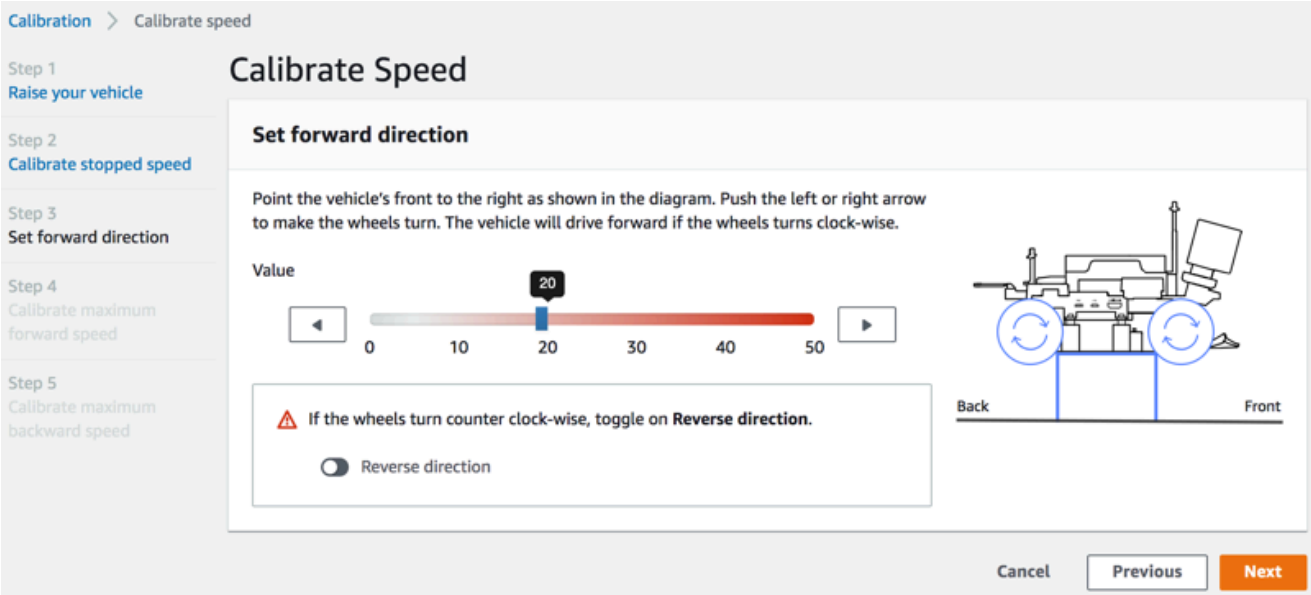
b. 정지 속도를 보정하려면 왼쪽 또는 오른쪽 화살표를 눌러 바퀴의 회전이 멈출 때까지 장치 제어 콘솔의 정지 속도에서 정지 값을 점진적으로 변경합니다. Next(다음)를 선택합니다.



> **Note**
>
> 잡음이 들리기 시작할 때까지 정지 값을 왼쪽 또는 오른쪽으로 계속 누르면 바퀴가 움직이기 시작합니다. 따라서 이상적인 정지 속도 지점은 두 값의 중앙 값입니다. 예를

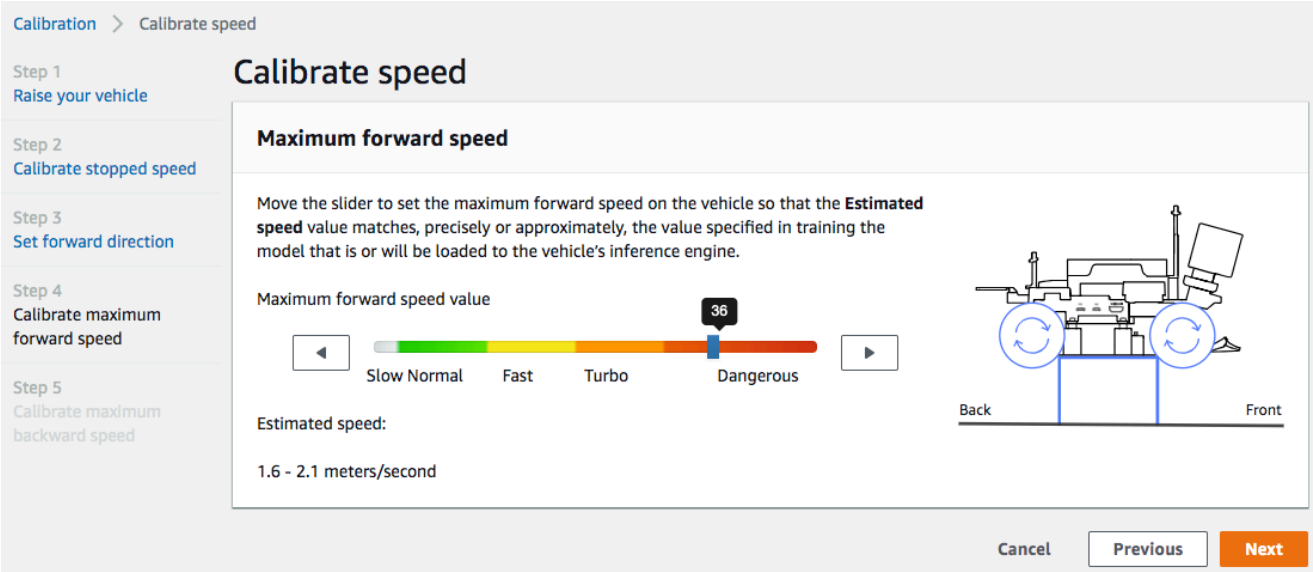
들어 왼쪽 16에서, 그리고 오른쪽 -4에서 잡음이 들리기 시작하면 최적의 정지 값은 10입니다.

c. 차량의 순방향을 설정하려면 차량을 화면 또는 아래 이미지와 같이 놓고 왼쪽/오른쪽 화살표를 눌러 바퀴를 회전시킵니다. 바퀴가 시계 방향으로 회전하면 순방향이 설정된 것입니다. 그렇지 않다면 역방향을 전환합니다. Next(다음)를 선택합니다.



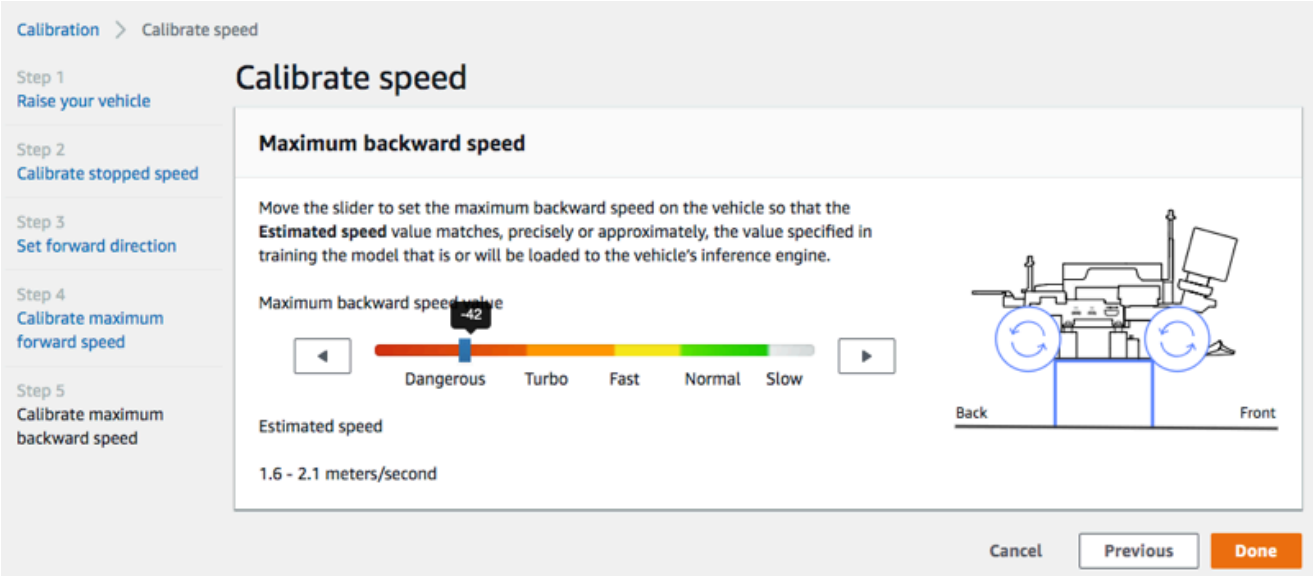
> **Note**
>
> AWS re:Invent 2018에서 배포된 차량은 전진 방향이 역방향으로 설정될 수 있습니다. 이때는 반드시 역방향을 전환해야 합니다.

d. 최대 전진 속도를 보정하려면 최대 전진 속도에서 슬라이더를 왼쪽 또는 오른쪽으로 가볍게 움직여 예상 속도 값이 시뮬레이션에 지정된 최대 속도와 같거나 비슷하도록 최대 전진 속도 값 수치를 양수 값으로 점진적으로 조정합니다. Next(다음)를 선택합니다.



> **Note**
>
> 차량의 실제 최대 속도는 트랙 표면의 마찰력과 차량 배터리 레벨에 따라 다릅니다. 따라서 차량의 스로틀 제한을 시뮬레이션 훈련에서 지정하는 최대 속도보다 20~30% 높게 설정하면 유연한 보정이 가능합니다. 일반적으로 최대 속도 값은 녹색 영역에서 벗어나지 않도록 설정해야 합니다. 이 범위를 벗어나면 차량이 너무 빠르게 주행할 가능성이 높아져 파손 위험이 큽니다. 또한 훈련 행동 공간은 2m/s보다 큰 최대 속도를 지원하지 않습니다.

e. 최대 후진 속도를 보정하려면 최대 후진 속도에서 슬라이더를 왼쪽 또는 오른쪽으로 가볍게 움직여 예상 속도 값이 시뮬레이션에 지정된 최대 속도와 같거나 비슷하도록 최대 후진 속도 값 숫자를 음수 값으로 점진적으로 조정합니다. 완료를 선택합니다.



> **Note**
>
> AWS DeepRacer 차량은 자율 주행 모드에서 역방향 속도를 사용하지 않습니다. 역방향 속도는 차량의 수동 주행 모드를 편안하게 제어할 수만 있다면 어떠한 값으로도 설정할 수 있습니다.

이것으로 AWS DeepRacer 차량의 최대 속도 보정이 완료됩니다.

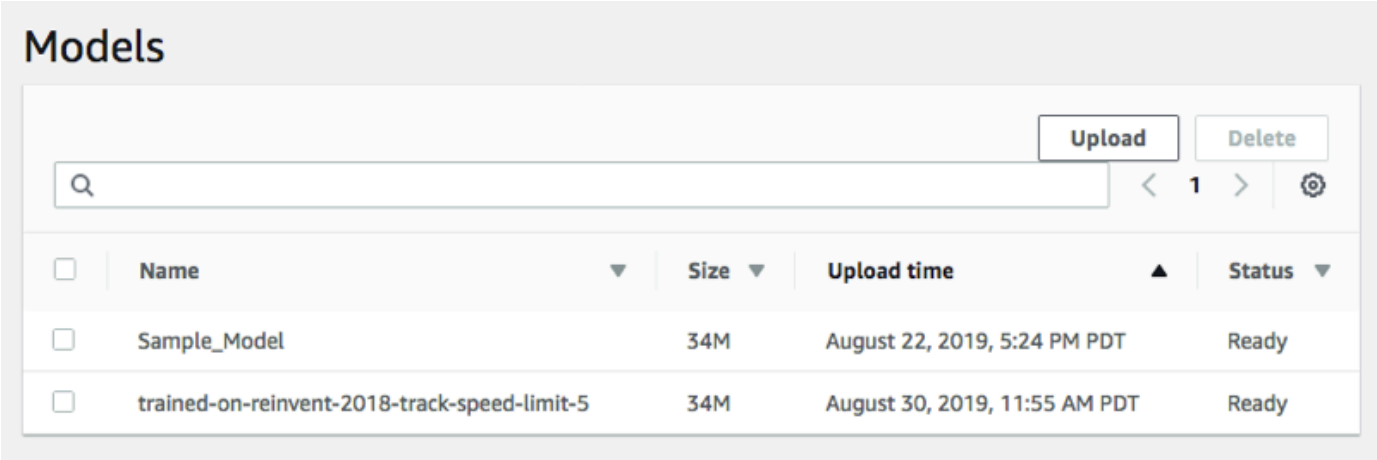
AWS DeepRacer 차량에 모델 업로드

AWS DeepRacer 차량 자율 주행을 시작하려면 먼저 1개 이상의 AWS DeepRacer 모델을 AWS DeepRacer 차량에 업로드해야 합니다.

또한 모델을 업로드하려면 먼저 모델을 훈련 및 평가해야 합니다. 모델 훈련은 AWS DeepRacer 콘솔에서 가능합니다. 훈련을 마치면 모델 아티팩트를 Amazon S3 스토리지에서 컴퓨터에서 액세스할 수 있는 (로컬 또는 네트워크) 드라이브로 다운로드해야 합니다.

훈련된 모델을 차량에 업로드하는 방법

1. 디바이스 콘솔의 기본 탐색 창에서 모델을 선택합니다.



2. 모델 페이지의 모델 목록 위에서 업로드를 선택합니다.
3. 파일 선택기에서 모델 아티팩트를 다운로드한 드라이브 또는 공유 폴더로 이동하고, 업로드할 압축된 모델 파일(`*.tar.gz` 확장자)을 선택합니다.

   제대로 업로드된 모델만 모델 목록에 추가되며, 자율 주행 모드에서 이를 차량의 추론 엔진에 로드할 수 있습니다. 모델을 차량의 추론 엔진에 로드하는 방법에 대한 지침은 AWS DeepRacer 차량 자율 주행 단원을 참조하십시오.

AWS DeepRacer 차량을 운전해 보십시오.

AWS DeepRacer 차량 설정 후에는 차량을 수동으로 주행하기 시작하거나 차량의 디바이스 콘솔을 사용하여 자율적으로 주행하도록 둘 수 있습니다.

자율 주행을 하려면 AWS DeepRacer 모델을 훈련하고, 훈련된 모델 아티팩트를 차량에 배포해야 합니다. 자율 레이싱 모드에서 추론 엔진에서 실행 중인 모델은 차량의 주행 방향과 속도를 제어합니다. 훈련된 모델을 차량으로 다운로드하지 않은 상태에서는 차량의 디바이스 콘솔을 사용하여 차량을 수동으로 운전할 수 있습니다.

자율 주행 시 차량의 성능에 영향을 미치는 요인들이 많습니다. 훈련된 모델과 차량 보정을 비롯해 표면 마찰, 색상 대비, 조명 반사 같은 트랙 조건도 여기에 포함됩니다. 차량이 최적의 성능을 발휘하려면 시뮬레이션에서 실제 환경으로 이어지는 모델 전환이 정확하고, 적합하고, 유의적이어야 합니다. 자세한 내용은 the section called “실제 환경에 대한 훈련 최적화” 섹션을 참조하십시오.

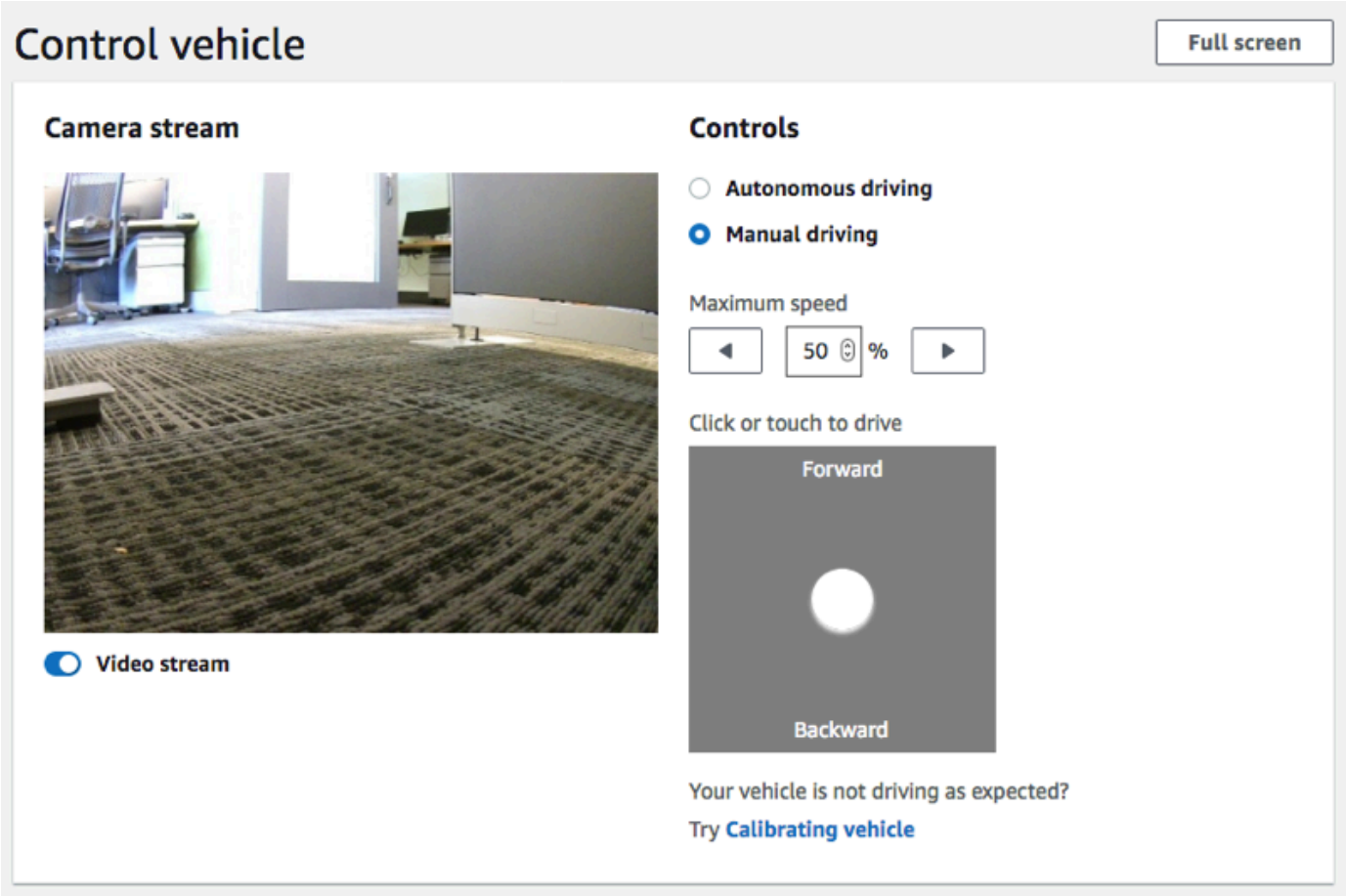
AWS DeepRacer 차량 수동 주행

훈련된 모델이 없거나, 혹은 훈련된 모델을 AWS DeepRacer 차량에 배포하지 않았다면 자율 주행이 어렵습니다. 하지만 수동으로 운전할 수는 있습니다.

AWS DeepRacer 차량을 수동으로 주행하려면 아래 단계를 따릅니다.

AWS DeepRacer 차량을 수동으로 주행하는 방법

1. AWS DeepRacer 차량이 Wi-Fi 네트워크에 연결된 상태에서 지침에 따라 차량의 디바이스 제어 콘솔에 로그인합니다.
2. 차량 제어 페이지의 제어에서 수동 운전을 선택합니다.



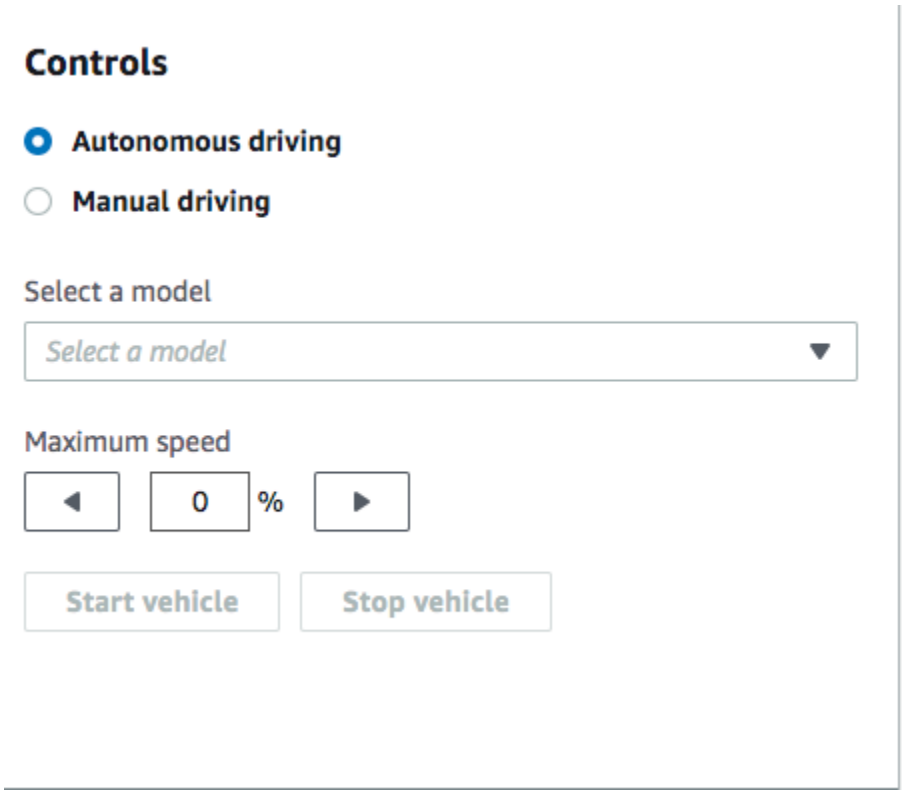
3. 클릭 또는 터치하여 주행 시작 아래에서 주행 패드 내 위치를 클릭하거나 터치하여 차량을 주행합니다. 차량의 전방 카메라에서 수집한 이미지가 카메라 스트림 아래 비디오 플레이어에 표시됩니다.
4. 차량을 운전하는 동안 디바이스 콘솔에서 비디오 스트림을 켜거나 끄려면 카메라 스트림 화면에서 비디오 스트림 옵션을 전환합니다.
5. 3단계부터 반복하여 차량을 다른 위치로 주행합니다.

AWS DeepRacer 차량 자율 주행

자율 주행을 시작하려면 차량을 물리적 트랙 위에 놓고 다음과 같이 따릅니다.

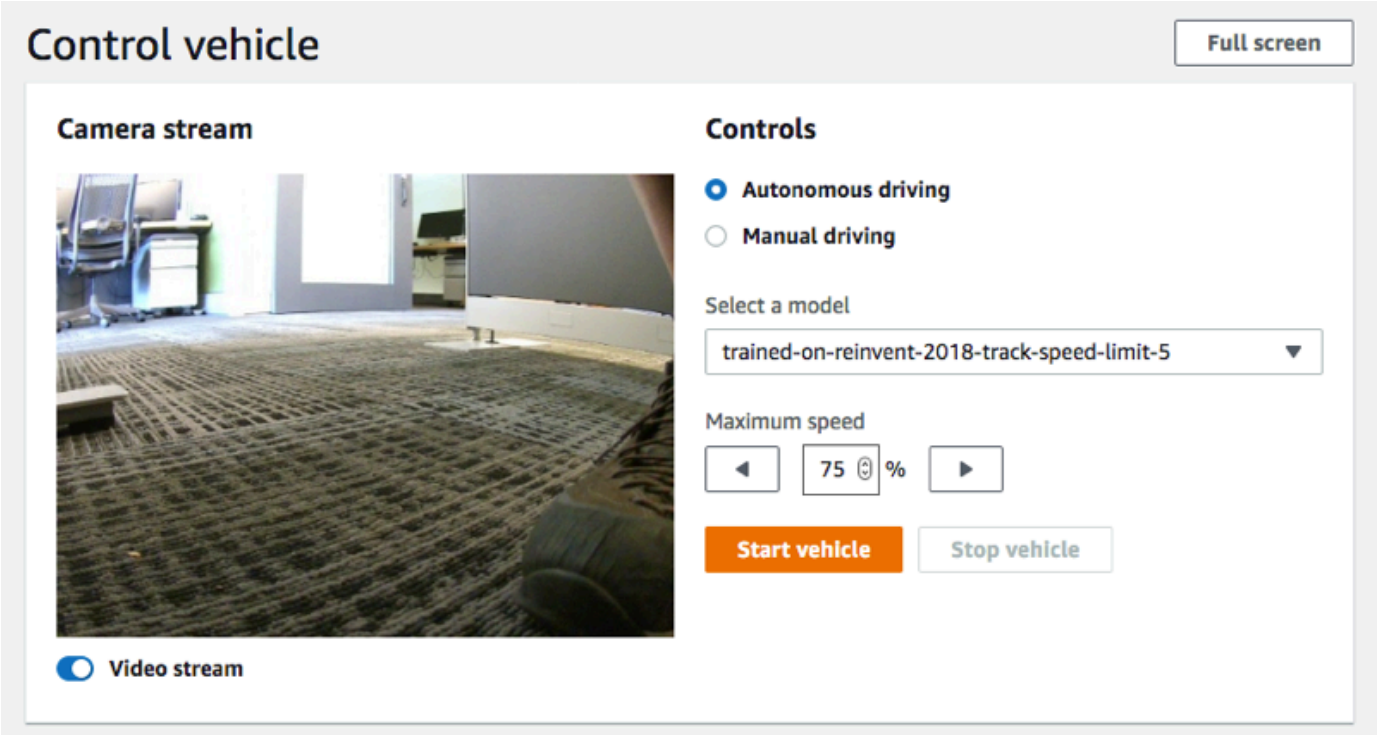
AWS DeepRacer 차량을 자율 주행하는 방법

1. 지침에 따라 차량의 디바이스 콘솔에 로그인한 후 다음과 같은 방법으로 자율 주행을 시작합니다.
2. 차량 제어 페이지의 제어에서 자율 주행을 선택합니다.



3. 모델 선택 드롭다운 목록에서 업로드된 모델을 선택합니다. 모델 로드를 선택합니다. 그러면 모델이 추론 엔진에 로드되기 시작합니다. 로딩 프로세스는 약 10초 걸립니다.
4. 차량의 최대 속도 설정을 모델 훈련 시 사용한 최대 속도의 비율로 조정합니다.

   이때 차량의 최대 속도는 실제 트랙의 표면 마찰 등 몇 가지 요인으로 인해 훈련 시 사용한 최대 속도보다 느릴 수 있습니다. 최적의 설정을 찾으려면 실험이 필요합니다.



5. 차량 출발을 선택하여 차량을 자율 주행합니다.
6. 차량을 운전하는 동안 디바이스 콘솔에서 비디오 스트림을 켜거나 끄려면 카메라 스트림 화면에서 비디오 스트림 옵션을 전환합니다.
7. 물리적 트랙에서 또는 디바이스 콘솔의 스트리밍 비디오 플레이어에서 차량이 주행하는 모습을 살펴봅니다.
8. 차량을 멈추려면 차량 정지을 선택합니다.

동일한 모델이나 다른 모델로 3단계부터 주행을 반복합니다.

AWS DeepRacer 차량 설정 검사 및 관리

초기 설정 이후 AWS DeepRacer 디바이스 제어 콘솔을 사용해 차량의 설정을 관리할 수 있습니다. 작업에는 다음이 포함됩니다.

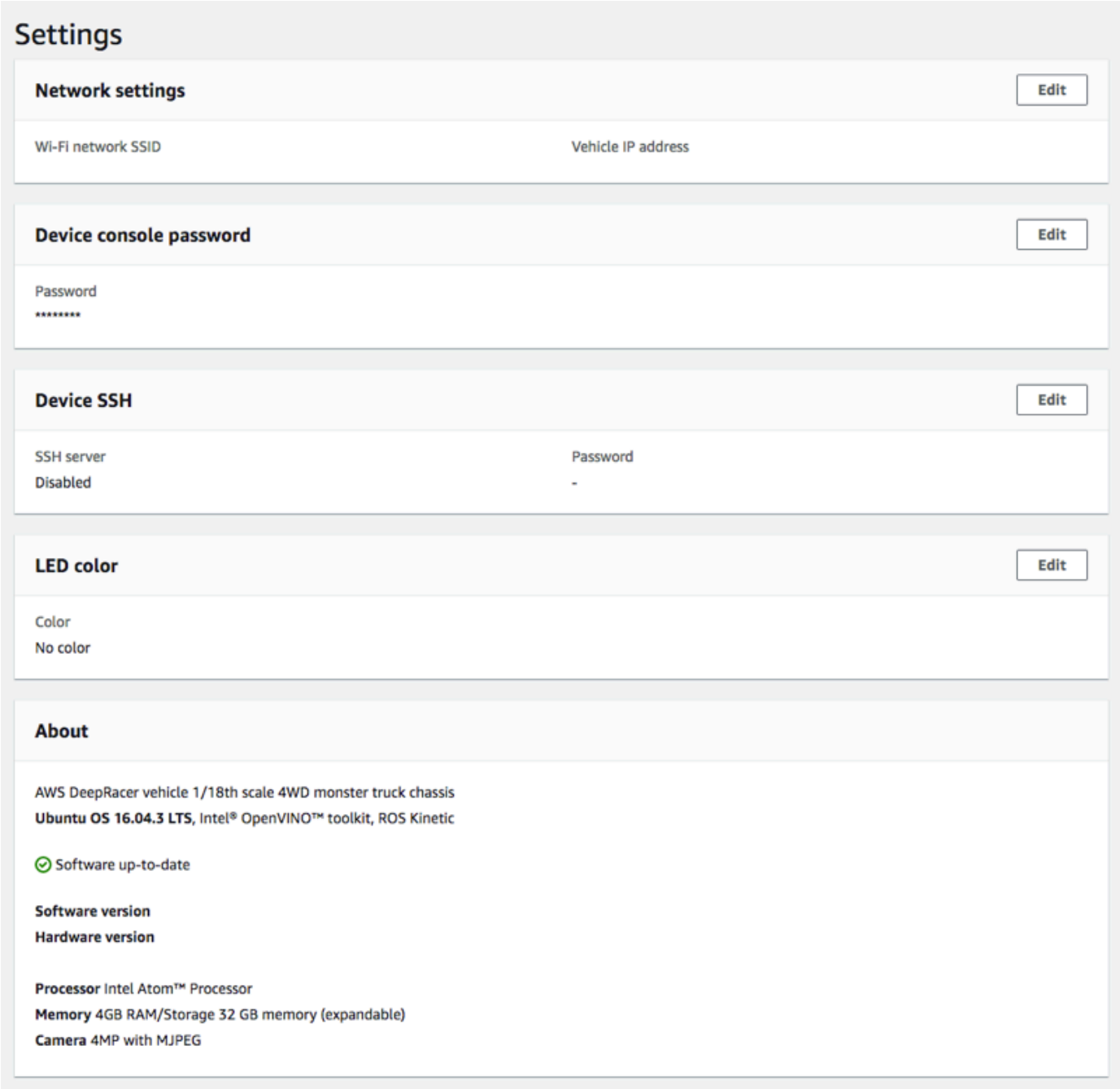
- 다른 Wi-Fi 네트워크 선택
- 디바이스 콘솔 암호 재설정
- 디바이스 SSH 설정 활성화 또는 비활성화
- 차량의 미등 LED 색상 구성
- 디바이스 소프트웨어 및 하드웨어 버전 검사

- 차량 배터리 수준 확인

아래 절차에서는 위의 작업을 안내합니다.

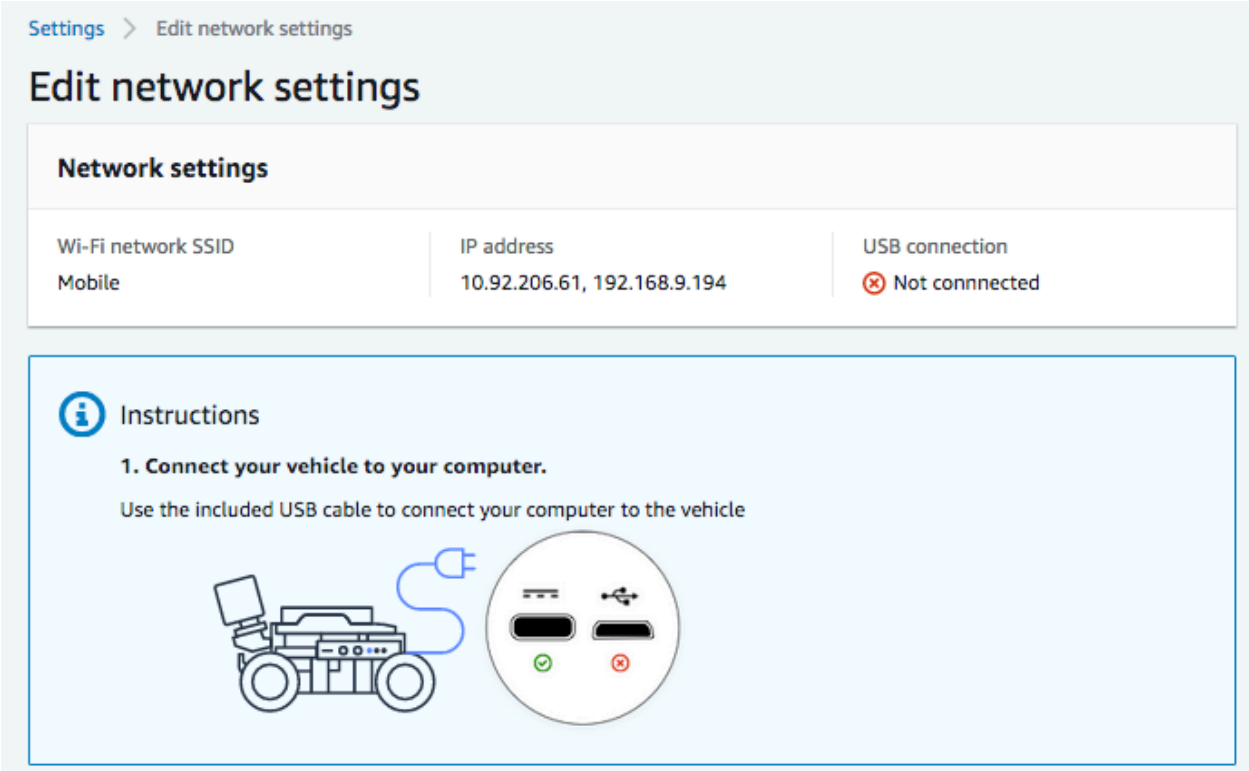
차량 설정을 검사하고 관리하려면

1. AWS DeepRacer 차량이 Wi-Fi 네트워크에 연결된 상태에서 지침에 따라 차량의 디바이스 제어 콘솔에 로그인합니다.
2. 기본 탐색 창에서 설정을 선택합니다.
3. 설정 페이지에서 다음 중 원하는 작업 하나를 실행합니다.



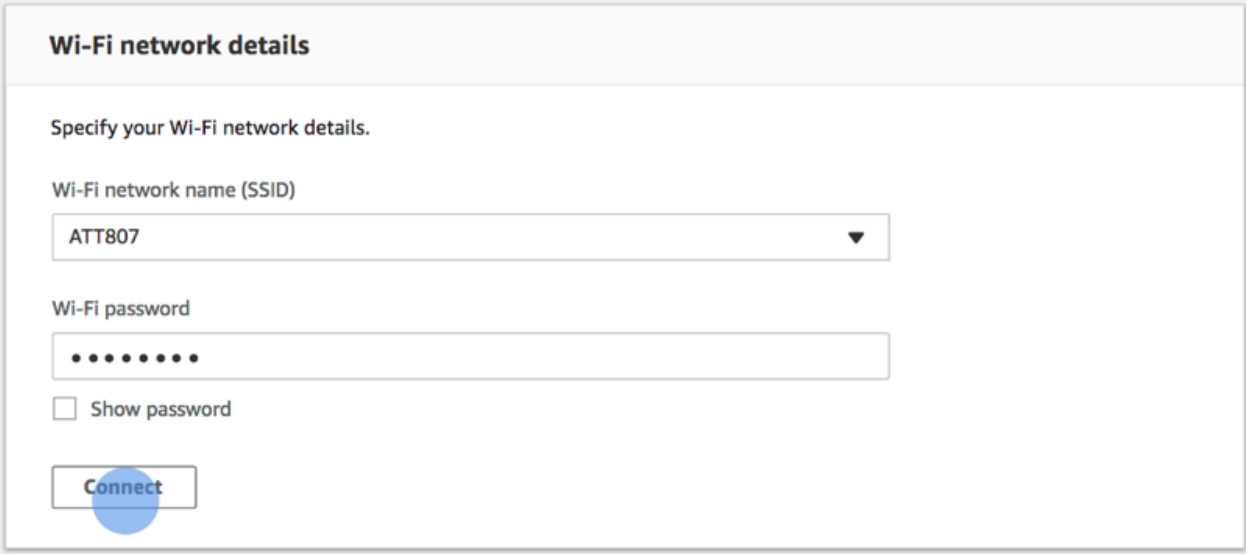
a. 다른 Wi-Fi 네트워크를 선택하려면 네트워크 설정에서 편집을 선택한 다음 아래 단계를 따르십시오.

   i. 네트워크 설정 편집에서 보이는 것처럼 지침에 따라 USB-USB-C 케이블을 사용해 차량을 컴퓨터에 연결합니다. USB 연결 상태가 연결 상태가 되면 deepracer.aws로 이동 버튼을 선택하여 디바이스 콘솔 로그인 페이지를 엽니다.



ii. 디바이스 콘솔 로그인 페이지에서 차량 하단에 인쇄된 암호를 입력하고 차량 액세스을 선택합니다.

iii. Wi-Fi 네트워크 세부 정보 아래 있는 드롭다운 목록에서 Wi-Fi 네트워크를 선택하여 해당 네트워크의 암호를 입력한 후 연결을 선택합니다.



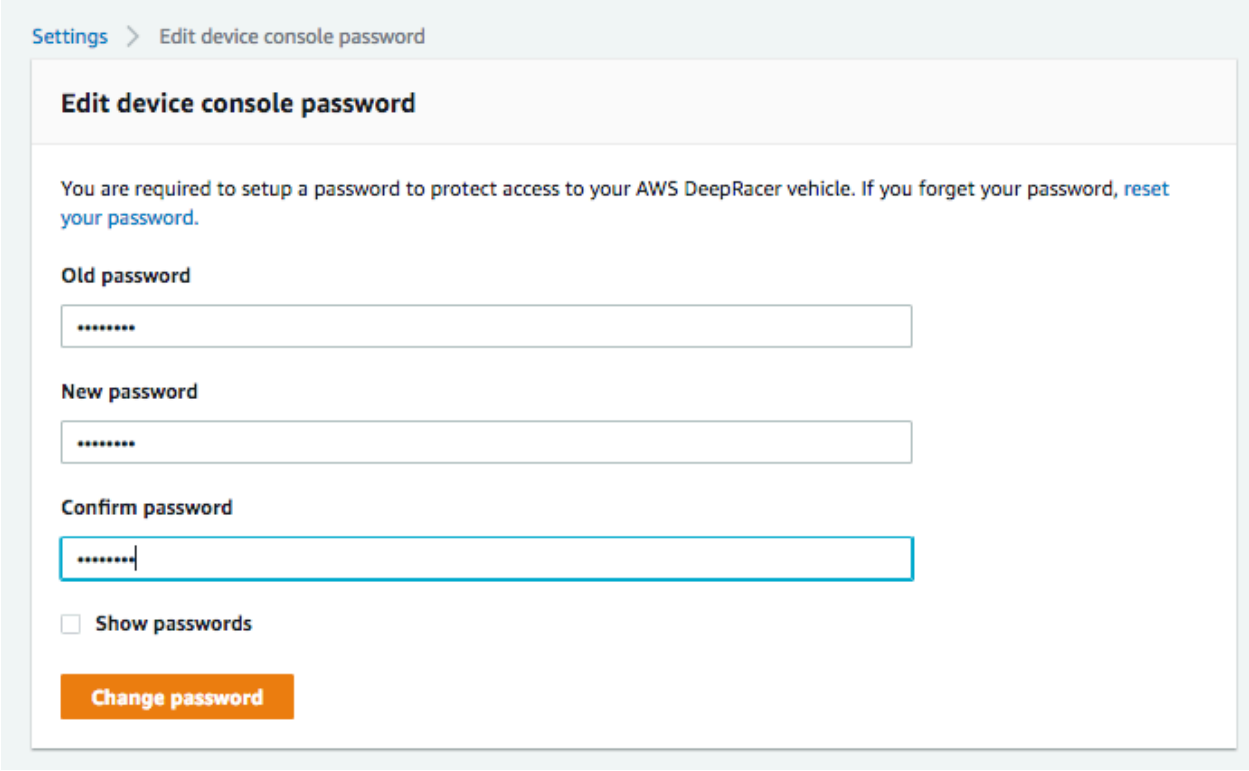
iv. Wi-Fi 연결을 나타내는 차량 상태가 연결 상태가 되면 다음을 선택하여 디바이스 콘솔의 설정 페이지로 돌아갑니다. 그러면 새로운 차량 IP 주소가 표시됩니다.

b. 디바이스 콘솔에 로그인하기 위한 암호를 재설정하려면 디바이스 콘솔 암호 편집을 선택한 다음 아래 단계를 따르십시오.

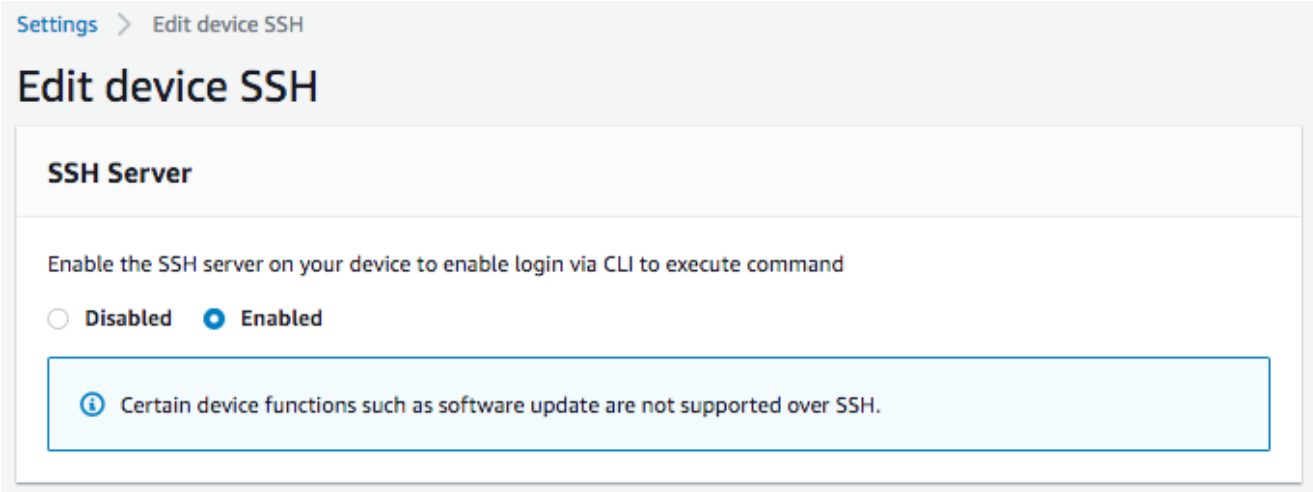
   i. 디바이스 콘솔 암호 편집 페이지에서 새 비밀번호에 새로운 암호를 입력합니다.

   ii. 암호 확인에 새로운 암호를 다시 입력하여 변경 여부를 확정합니다. 계속 하려면 이때 입력하는 암호 값이 동일해야 합니다.

   iii. 비밀번호 변경을 선택하여 작업을 완료합니다. 이 옵션은 위 단계에서 유효한 암호 값을 입력하여 확인한 경우에만 활성화됩니다.

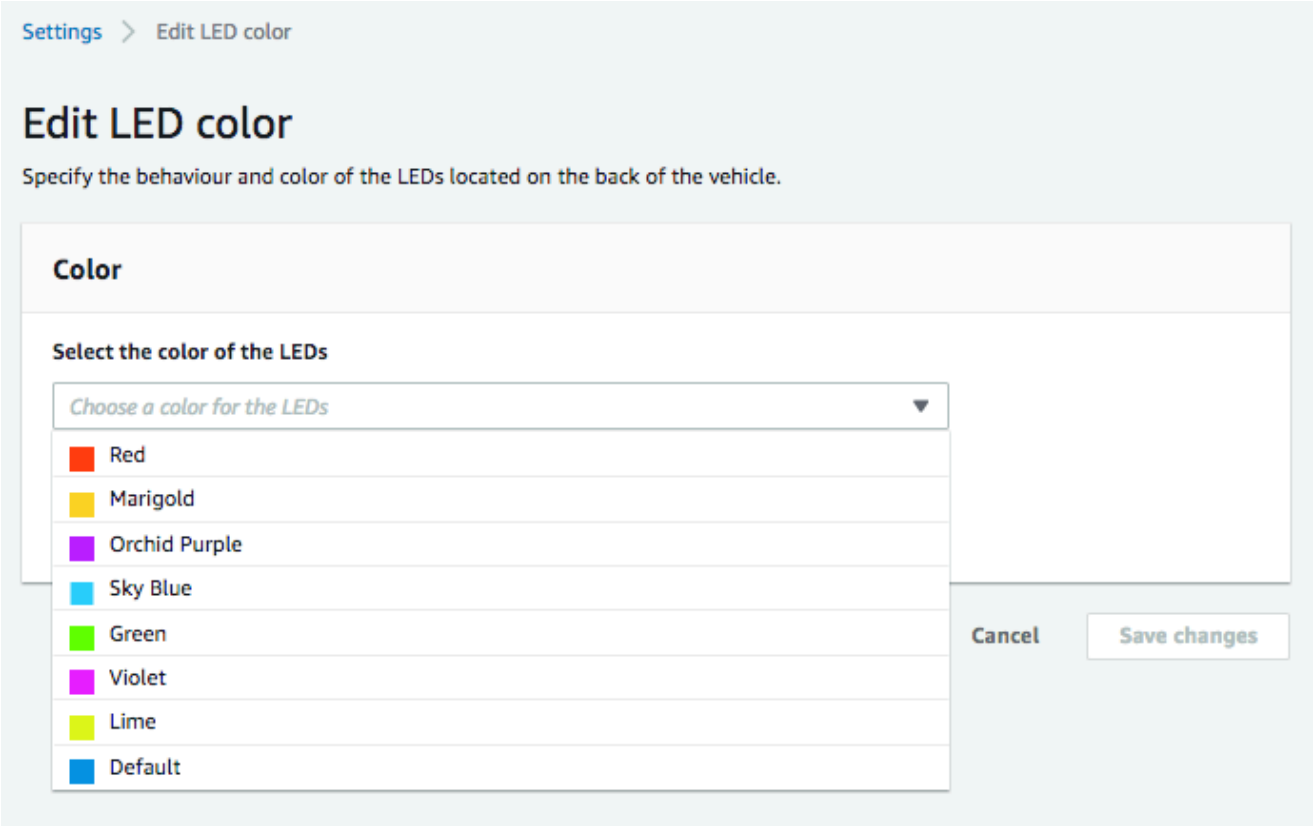


c. 차량에 대한 SSH 연결을 활성화하거나 비활성화하려면 디바이스 SSH에 대해 편집을 선택한 다음 활성화 또는 비활성화를 선택합니다.



4. 트랙에서 차량을 구별하기 쉽도록 차량의 미등 LED 색상을 변경하려면 설정 페이지에서 LED 색상에 대해 편집을 선택하고 다음과 같이 하십시오.

   a. LED 색상 편집 페이지의 LED 색상 선택 드롭다운 목록에서 사용 가능한 색상을 선택합니다.



   동시에 트랙을 주행하는 다른 차량과 자신의 차량을 식별하는 데 도움이 될 수 있는 색상을 선택해야 합니다.

   b. 변경 사항 저장을 선택하여 작업을 완료합니다.

변경 사항 저장 기능은 색상을 선택해야만 활성화됩니다.

5. 디바이스의 소프트웨어 및 하드웨어 버전을 검사하고 시스템 및 카메라 구성을 알아보려면 설정 아래의 정보 섹션을 확인합니다.
6. 차량의 배터리 충전 수준을 검사하려면 기본 탐색 창의 아래쪽 부분을 확인합니다.

AWS DeepRacer 차량 로그 보기

AWS DeepRacer 차량은 작업 이벤트를 기록하기 때문에 차량 작동 시 발생한 문제를 해결하는 데 유용합니다. AWS DeepRacer 차량 로그는 다음과 같이 두 가지 유형이 있습니다.

- 시스템 이벤트 로그는 프로세스 관리, Wi-Fi 연결, 암호 재설정 이벤트 등 차량의 컴퓨터 운영 체제에서 발생하는 작업을 추적합니다.
- 로봇 운영 체제 로그는 차량 주행, 비디오 스트리밍, 정책 추론 작업 등 로봇 작업 시 차량의 운영 체제 노드에서 일어나는 작업 상태를 기록합니다.

다음과 같은 방법으로 디바이스 로그를 확인합니다.

1. AWS DeepRacer 차량이 Wi-Fi 네트워크에 연결된 상태에서 지침에 따라 차량의 디바이스 제어 콘솔에 로그인합니다.
2. 디바이스 콘솔의 기본 탐색 창에서 로그를 선택합니다.
3. 시스템 이벤트를 보려면 시스템 이벤트 로그에 있는 이벤트 목록을 아래로 내리면서 확인합니다.

System event log

```
Apr 8 15:16:07 amss-42im login: message repeated 2 times: [ <INFO> Status returned from login proxy: 200]
Apr 8 15:16:07 amss-42im wifi_settings: <INFO> Check OTG Link State: not connected
Apr 8 15:16:07 amss-42im wifi_settings: <INFO> host: https://10.92.206.61/home otg_connected: not connected is_usb_connected: not
connected
Apr 8 15:16:07 amss-42im login: <INFO> Status returned from login proxy: 200
Apr 8 15:16:07 amss-42im login: message repeated 2 times: [ <INFO> Status returned from login proxy: 200]
Apr 8 15:16:07 amss-42im vehicle_control: <INFO> Changed the vehicle state to auto
Apr 8 15:16:07 amss-42im login: <INFO> Status returned from login proxy: 200
Apr 8 15:16:07 amss-42im wifi_settings: <INFO> Check OTG Link State: not connected
Apr 8 15:16:08 amss-42im utility: <INFO> Command executing: hostname -I
Apr 8 15:16:08 amss-42im utility: <INFO> ['10.92.206.61 192.168.9.194 ', '']
Apr 8 15:16:11 amss-42im login: <INFO> Status returned from login proxy: 200
Apr 8 15:16:41 amss-42im login: message repeated 3 times: [ <INFO> Status returned from login proxy: 200]
Apr 8 15:16:41 amss-42im ssh_api: <INFO> Providing ssh enabled as response
Apr 8 15:16:41 amss-42im utility: <INFO> Command executing: /bin/systemctl --no-pager status ssh
Apr 8 15:16:41 amss-42im wifi_settings: <INFO> Check OTG Link State: not connected
Apr 8 15:16:41 amss-42im utility: <INFO> ● ssh.service - OpenBSD Secure Shell server#012 Loaded: loaded (/lib/systemd/system/ssh.service;
enabled; vendor preset: enabled)#012 Active: active (running) since Fri 2019-04-05 15:43:20 EDT; 2 days ago#012 Main PID: 16466 (sshd)#012
```

4. 로봇 운영 체제 이벤트를 보려면 로봇 운영 체제 로그에 있는 이벤트 목록을 아래로 내리면서 확인합니다.

Robot operating system log

```
1554750920.064320544 Node Startup
1554750920.131309136 INFO [/opt/workspace/AwsSilverstoneDeviceLib/ros-src/servo_pkg/src/servo_node.cpp:439(LedMgr::LedMgr) [topics:
/rosout] LedMgr pwm channel creation
1554750920.201161384 INFO [/tmp/binarydeb/ros-kinetic-roscpp-1.12.14/src/libros/service.cpp:80(service::exists) [topics: /rosout]
waitForService: Service [/media_state] has not been advertised, waiting...
1554750920.640698003 INFO [/tmp/binarydeb/ros-kinetic-roscpp-1.12.14/src/libros/service.cpp:122(service::waitForService) [topics: /rosout]
waitForService: Service [/media_state] is now available.
1554750920.578106989 INFO [/opt/workspace/AwsSilverstoneDeviceLib/ros-src/web_video_server
/src/web_video_server.cpp:96(WebVideoServer::spin) [topics: /rosout] Waiting For connections on 0.0.0.0:8080
1554750921.752294063 INFO [navigation_node.py:154(set_action_space_scales) [topics: /auto_drive, /rosout, /rl_results] Action space scale set:
{'steering_max': 30.0, 'speed_max': 0.8}
Mapping equation params a: -1.875 b: 2.75
1554750930.167246103 INFO [software_update_process.py:25(logger) [topics: /rosout] /software_update: [04/08/19 15:15:30] Setup Ethernet
over OTG.
1554750930.174333095 INFO [software_update_process.py:25(logger) [topics: /rosout] /software_update: [04/08/19 15:15:30] Entering
daemon loop.
1554750930.205965042 INFO [software_update_process.py:25(logger) [topics: /rosout] /software_update: [04/08/19 15:15:30] Updating
network information.
1554750930.209075927 INFO [software_update_process.py:25(logger) [topics: /rosout] /software_update: [04/08/19 15:15:30] Checking
software update...
1554750938.287539958 INFO [software_update_process.py:25(logger) [topics: /rosout] /software_update: [04/08/19 15:15:38] Verifying
package aws-deepracer-core...
```

AWS DeepRacer 디바이스 업데이트 및 복원

AWS DeepRacer 디바이스를 Ubuntu 20.04 Focal Fossa, Intel® OpenVINO™ 툴킷 2021.1.110, ROS2 Foxy Fitzroy 및 Python 3.8 등의 최신 소프트웨어 스택으로 업데이트하십시오. 이 업데이트는 AWS DeepRacer 오픈 소스 프로젝트를 실행하는 데 필요하지만 그 외에는 선택 사항입니다. AWS DeepRacer는 Ubuntu 20.04 Focal Fossa와 ROS2 Foxy Fitzroy만 지원합니다.

> **Important**
>
> 새 AWS DeepRacer 소프트웨어 스택으로 업데이트하면 AWS DeepRacer 디바이스의 모든 데이터가 삭제됩니다.

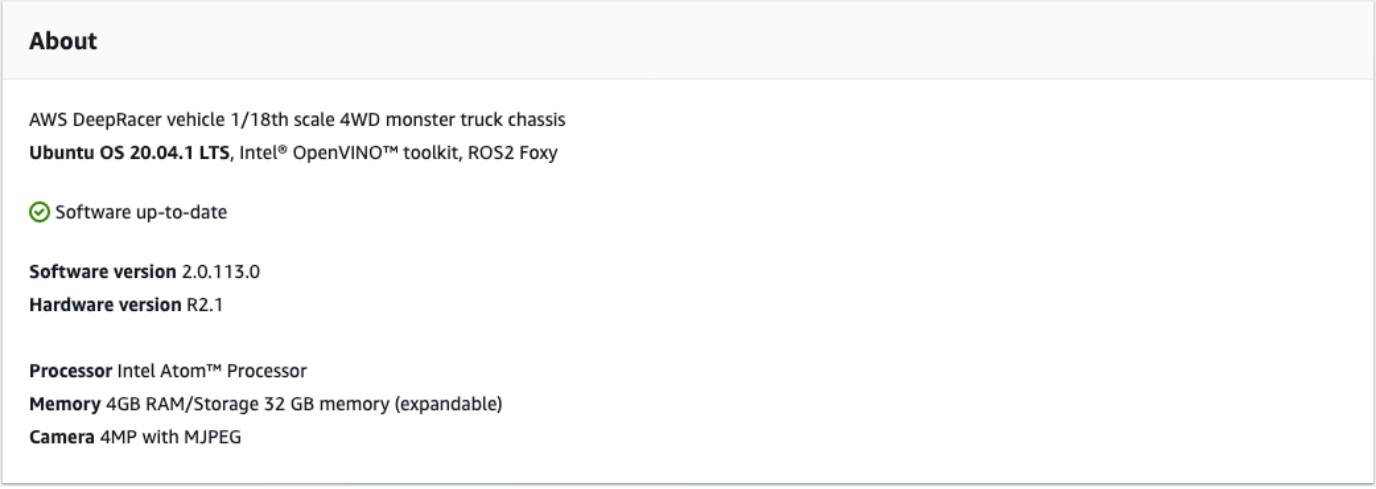
주제

- AWS DeepRacer 디바이스에서 현재 실행 중인 소프트웨어 버전을 확인하십시오.
- AWS DeepRacer 디바이스를 Ubuntu 20.04 소프트웨어 스택으로 업데이트할 준비를 하십시오.
- AWS DeepRacer 디바이스를 Ubuntu 20.04 소프트웨어 스택으로 업데이트하십시오.

AWS DeepRacer 디바이스에서 현재 실행 중인 소프트웨어 버전을 확인하십시오.

AWS DeepRacer 디바이스에서 현재 실행 중인 소프트웨어 버전을 확인하려면

1. AWS DeepRacer 디바이스 콘솔에 로그인합니다. 방법을 알아보려면 the section called “디바이스 콘솔 시작”의 단계를 따르십시오.
2. 탐색 창에서 설정을 선택합니다.
3. 정보 섹션을 확인하여 AWS DeepRacer 차량이 현재 실행 중인 소프트웨어 버전을 확인하십시오.



AWS DeepRacer 디바이스를 Ubuntu 20.04 소프트웨어 스택으로 업데이트할 준비를 하십시오.

이 주제에서는 AWS DeepRacer Ubuntu 설치 미디어를 생성하는 프로세스를 안내합니다. 부팅 가능한 USB 드라이브를 준비하려면 추가 하드웨어가 필요합니다.

사전 조건

시작하기 전에 다음 항목을 준비해야 합니다.

- AWS DeepRacer 디바이스
- USB 플래시 드라이브 1개, 32GB 이상
- 사용자 지정 AWS DeepRacer Ubuntu ISO 이미지.
- 최신 AWS DeepRacer 소프트웨어 업데이트 패키지.
- 운영 체제와 호환되는 UNetbootin의 사본.
- USB 설치 미디어를 준비하기 위해 Ubuntu, Windows 또는 macOS를 실행하는 컴퓨터 HDMI 타입 A 케이블로 마우스, 키보드 및 모니터를 연결하여 AWS DeepRacer 디바이스의 컴퓨팅 모듈을 Linux 컴퓨터처럼 사용할 수도 있습니다.

준비

AWS DeepRacer 업데이트 미디어를 준비하려면 다음 작업을 수행합니다.

- USB 드라이브를 다음 2개 파티션으로 포맷합니다.
  - 4GB, FAT32 부트 파티션
  - 최소 18GB의 NTFS 데이터 파티션
- 재부팅 시 업데이트를 시작하도록 USB 드라이브를 부팅 가능하도록 설정합니다.
  - 필요한 사용자 지정 Ubuntu ISO 이미지를 부트 파티션으로 굽습니다.
  - 필요한 업데이트 파일을 USB 드라이브의 데이터 파티션에 복사합니다.

부팅 가능한 USB 드라이브 준비

다음 지침에 따라 Ubuntu(리눅스), 윈도우 또는 macOS에서 AWS DeepRacer 업데이트 미디어를 준비하십시오. 사용 중인 컴퓨터에 따라 특정 작업은 운영 체제마다 다를 수 있습니다. 사용 중인 운영 체제에 해당하는 탭을 선택합니다.

Ubuntu

여기의 지침에 따라 AWS DeepRacer 디바이스의 컴퓨팅 모듈을 포함한 Ubuntu 컴퓨터를 사용하여 AWS DeepRacer 디바이스용 업데이트 미디어를 준비하십시오. 다른 Linux 배포판을 사용하는 경우 `apt-get *` 명령을 운영 체제의 패키지 관리자와 호환되는 것으로 바꾸십시오.

USB 드라이브를 지우고 파티션을 나누려면

1. 다음 명령을 실행하여 GParted를 설치해 시작합니다.

   ```
   sudo apt-get update; sudo apt-get install gparted
   sudo gparted
   ```

2. USB 드라이브를 지우려면 디바이스 경로가 필요합니다. GParted 콘솔에서 찾아서 USB 드라이브를 지우려면 다음을 수행하십시오.

   a. 메뉴 막대에서 보기를 선택한 다음 디바이스 정보를 선택합니다. 선택한 디스크의 모델, 크기 및 경로를 보여주는 사이드바가 나타납니다.

   b. 메뉴 막대의 GParted로 이동하여 USB 드라이브를 선택한 다음 디바이스를 선택하고 마지막으로 목록에서 USB 드라이브를 선택합니다. 디바이스 설명에 표시된 크기 및 모델을 USB 드라이브와 일치시키십시오.

   c. 올바른 디스크를 선택했는지 확인했으면 기존 파티션을 모두 삭제하십시오.

      파티션이 잠긴 경우 콘텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 unmount(탑재 해제)를 선택합니다.

3. 4GB의 용량이 남아 있는 FAT32 파티션을 생성하려면 왼쪽 상단에서 파일 아이콘을 선택하고, 다음과 같이 파라미터를 설정한 다음 추가를 선택합니다.

   이전의 여유 공간: **1**

   새 용량:**4096**

   이후의 여유 공간: **<remaining size>**

   다음 단위로 맞춤: **MiB**

   다음으로 생성: **Primary Partition**

   파티션 이름:

   파일 시스템: **fat32**

   Label(레이블): **BOOT**
4. 18GB 이상의 용량이 남아 있는 NTFS 파티션을 생성하려면 파일 아이콘을 선택하고, 다음과 같이 파라미터를 설정한 다음 추가를 선택합니다.

   이전의 여유 공간: **0**

   새 용량:**<remaining size>**

   이후의 여유 공간: **0**

   다음 단위로 맞춤: **MiB**

   다음으로 생성: **Primary Partition**

   파티션 이름:

   파일 시스템: **nfts**

   Label(레이블): **Data**
5. 메뉴 표시줄에서 편집을 선택한 다음 모든 작업에 적용을 선택합니다. 변경 내용을 적용할지 묻는 경고 메시지가 나타납니다. 적용를 선택합니다.
6. FAT32 및 NTFS 파티션을 만든 후에는 USB 드라이브 파티션 정보가 GParted 콘솔에 나타납니다. 다음 단계를 완료하는 데 필요하므로 BOOT 파티션의 드라이브 경로를 기록해 둡니다.

USB 드라이브가 FAT32 파티션에서 부팅 가능하도록 하려면

1. 필수 조건 섹션에서 사용자 지정 Ubuntu ISO 이미지를 다운로드했는지 확인하십시오.
2. Ubuntu 20.04를 사용하는 경우 바이너리 파일을 사용하여 UNetbootin을 실행해야 합니다. 방법:

   a. 최신 UNetbootin 바이너리 파일을 다운로드 폴더에 다운로드하십시오. 이 예제에서는 `unetbootin-linux64-702.bin`을 사용합니다.

   b. Ctrl+Alt+T를 눌러 터미널 창을 엽니다. 또는 메뉴 막대에서 활동을 선택하고 검색 창에 **`terminal`**을 입력한 다음 터미널 아이콘을 선택합니다.

   c. 다음 명령을 사용하여 바이너리 파일 위치로 이동하고 파일에 실행 권한을 부여한 다음 UNetbootin을 실행합니다. 다운로드한 바이너리 파일의 버전과 버전이 일치하지 않는 경우 명령에서 파일 이름을 조정해야 합니다.

   ```
   cd Downloads
   sudo chmod +x ./unetbootin-linux64-702.bin
   sudo ./unetbootin-linux64-702.bin
   ```

   이전 Ubuntu 버전을 사용하는 경우 다음 명령을 실행하여 리포지토리에서 UNetbootin을 설치합니다.

   ```
   sudo add-apt-repository ppa:gezakovacs/ppa
   sudo apt-get update; sudo apt-get install unetbootin
   sudo unetbootin
   ```

3. UNetbootin 콘솔에서 다음을 수행합니다.

   a. 디스크 이미지 라디오 버튼을 선택합니다.

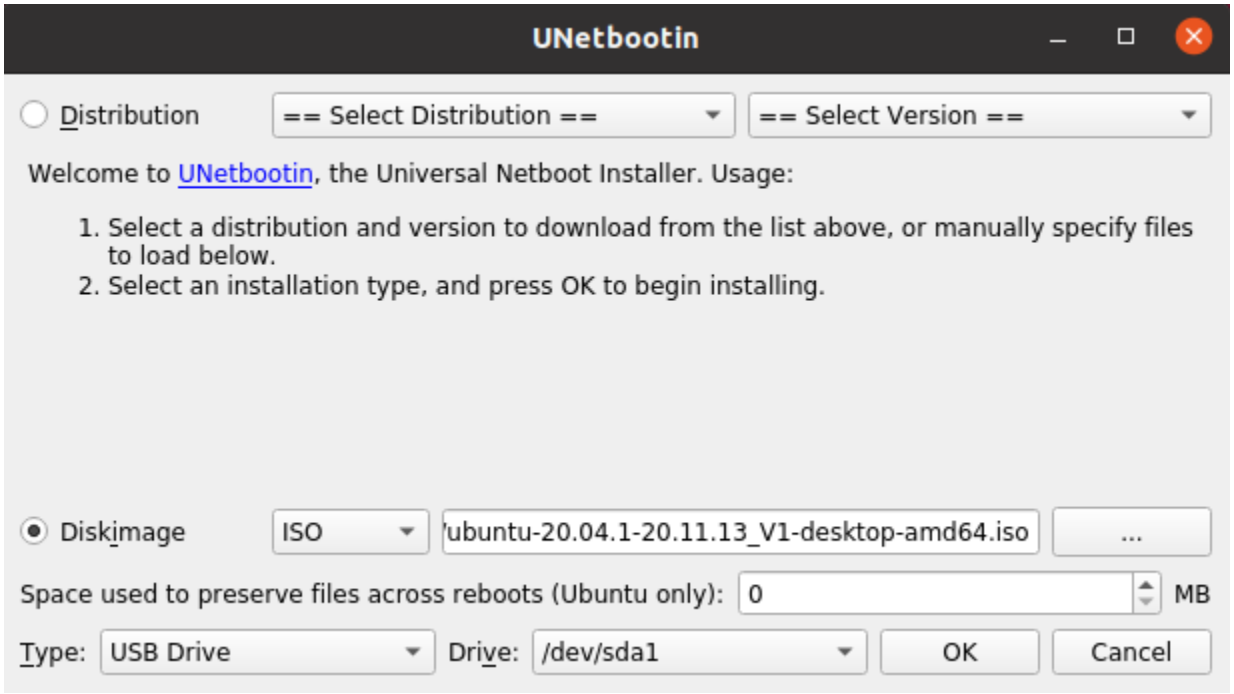
   b. 디스크 이미지 유형의 경우 드롭다운 메뉴에서 ISO를 선택합니다.

   c. 파일 선택기를 열고 사전 요구 사항 섹션에 제공된 Ubuntu ISO를 선택합니다.

   d. 유형에서 USB 드라이브를 선택합니다.

   e. 드라이브의 경우 BOOT 파티션의 드라이브 경로를 선택하십시오. 여기서는 **`/dev/sda1`**.

   f. 확인을 선택합니다.



> **Tip**
>
> /dev/sda1 탑재되지 않음 알림 메시지가 표시되면 확인를 선택하고, USB 드라이브를 뺐다가 다시 꽂은 다음 위의 단계에 따라 Ubuntu ISO 이미지를 생성합니다.

AWS DeepRacer 업데이트 파일을 NTFS 파티션으로 추출하려면

1. 사전 요구 사항 섹션에서 다운로드한 소프트웨어 업데이트 패키지의 압축을 풉니다.
2. 업데이트 패키지의 내용을 USB 드라이브의 데이터(NTFS) 파티션 루트에 추출합니다.

Windows

여기의 지침에 따라 Windows 컴퓨터를 사용하여 AWS DeepRacer 디바이스용 업데이트 미디어를 준비하십시오.

USB 드라이브를 지우려면

1. Windows 명령 프롬프트를 열고, `diskpart`를 입력한 다음 확인을 선택하여 Windows DiskPart를 시작합니다.

2. Microsoft DiskPart의 터미널이 열리면 사용 가능한 디스크를 나열하고 DISKPART> 프롬프트 다음에 `list disk`을 입력하여 정리할 USB 드라이브를 찾으십시오.
3. USB 드라이브에 해당하는 디스크를 선택합니다. 예를 들어 DISKPART> 프롬프트 다음에 `select Disk 2`을 입력했습니다. 다음 단계는 되돌릴 수 없으므로 출력을 주의 깊게 읽고 정리할 디스크를 선택했는지 확인하십시오.
4. 올바른 디스크를 선택했는지 확인했으면 DISKPART> 프롬프트 다음에 `Clean`를 입력합니다.
5. DISKPART> 프롬프트가 표시된 후 다시 `list disk`을 입력하십시오. 테이블에서 정리한 디스크를 찾아 디스크 크기를 여유 디스크 공간과 비교합니다. 두 값이 일치하면 정리에 성공한 것입니다.
6. DISKPART> 프롬프트 다음에 `Exit`를 입력하여 Windows DiskPart 콘솔을 종료합니다.

USB 드라이브를 파티셔닝하려면

1. Windows 명령 프롬프트를 열고, `diskmgmt.msc`를 입력한 다음 확인을 클릭하여 디스크 관리 콘솔을 시작합니다.
2. 디스크 관리 콘솔에서 USB 드라이브를 선택합니다.
3. 4GB 용량의 FAT32 파티션을 만들려면 USB 드라이브의 할당되지 않은 공간에서 컨텍스트 메뉴(오른쪽 클릭)를 열고 새 단순 볼륨을 선택합니다. 새로운 단순 볼륨 만들기 마법사가 나타납니다.
4. 새로운 단순 볼륨 만들기 마법사가 나타나면 다음을 수행합니다.

   a. 볼륨 크기 지정 페이지에서 다음 파라미터를 설정하고 다음을 선택합니다.

      단순 볼륨 크기(MB): **`4096`**

   b. 드라이브 문자 또는 경로 할당 페이지에서 다음 드라이브 문자 할당: 라디오 버튼을 선택하고 드롭다운 목록에서 드라이브 문자를 선택한 후 다음을 선택합니다. 할당된 드라이브 문자를 기록해 두십시오. 나중에 FAT32 파티션을 부팅할 수 있게 하려면 이 문자가 필요합니다.

   c. 파티션 포맷 페이지에서 다음 설정으로 이 볼륨 포맷 라디오 버튼을 선택하고 다음 파라미터를 설정한 후 다음을 선택합니다.

      파일 시스템: **`FAT32`**

      할당 단위 크기: **`Default`**

      볼륨 라벨: **`BOOT`**

빠른 포맷 수행을 체크된 상태로 두십시오.

5. 남은 디스크 용량으로 NTFS 파티션을 만들려면 USB 드라이브의 남은 할당되지 않은 공간에서 컨텍스트 메뉴(마우스 오른쪽 버튼 클릭)를 열고 새 단순 볼륨을 선택합니다. 새로운 단순 볼륨 만들기 마법사가 나타납니다.
6. 새로운 단순 볼륨 만들기 마법사가 나타나면 다음을 수행합니다.
   a. 볼륨 크기 지정 페이지에서 최대 디스크 공간(MB)과 일치하도록 단순 볼륨 크기(MB)를 설정한 후 다음을 선택합니다.
   b. 드라이브 문자 또는 경로 할당 페이지에서 다음 드라이브 문자 할당: 라디오 버튼을 선택하고 드롭다운 목록에서 드라이브 문자를 선택한 후 다음을 선택합니다.
   c. 파티션 포맷 페이지에서 다음 설정으로 이 볼륨 포맷 라디오 버튼을 선택하고 다음 파라미터를 설정한 후 다음을 선택합니다.

      파일 시스템: **`NTFS`**

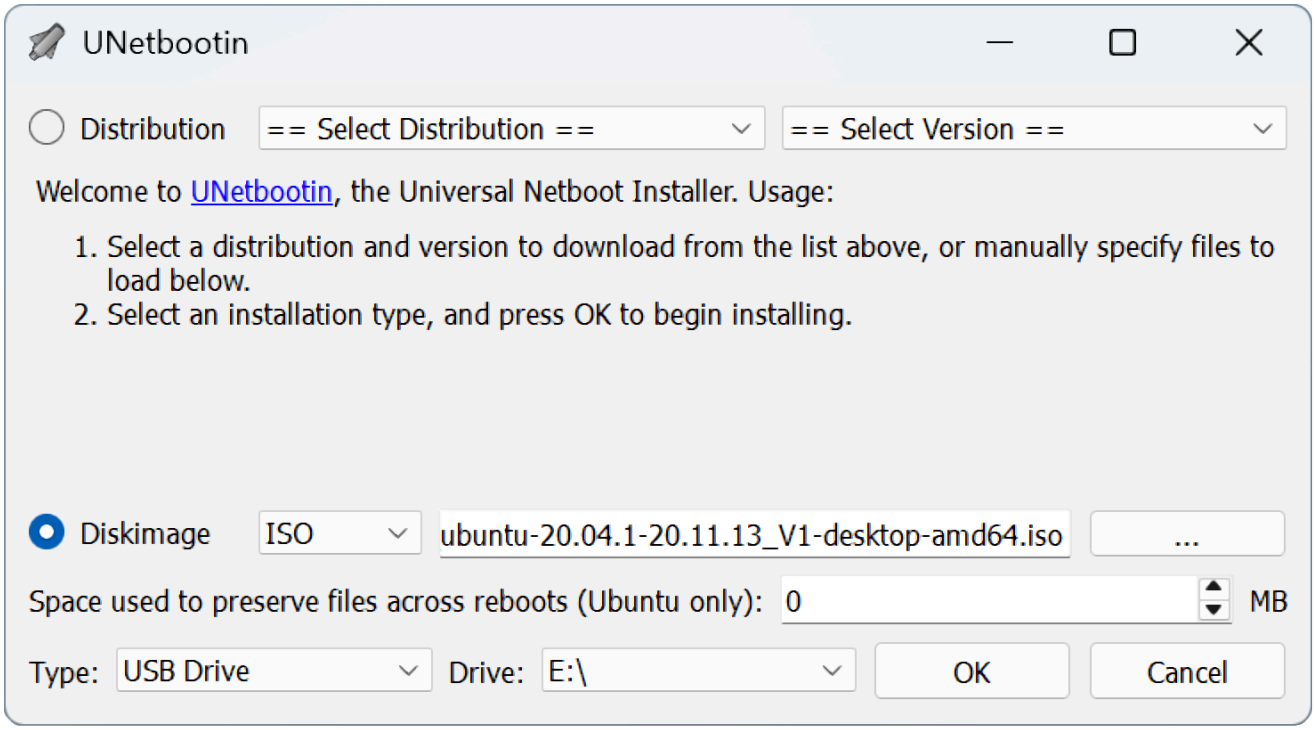
      할당 단위 크기: **`Default`**

      볼륨 라벨: **`Data`**

      빠른 포맷 수행을 체크된 상태로 두십시오.

USB 드라이브가 FAT32 파티션에서 부팅 가능하도록 하려면

1. 필수 조건 섹션에서 사용자 지정된 Ubuntu ISO 이미지를 다운로드했는지 확인하십시오.
2. UNetbootin을 다운로드한 후 UNetbootin 콘솔을 시작하십시오.
3. UNetbootin 콘솔에서 다음을 수행합니다.
   a. Disk image 라디오 버튼을 선택합니다.
   b. 디스크 이미지의 경우 드롭다운 목록에서 ISO를 선택합니다.
   c. File Picker를 연 다음 사용자 지정 Ubuntu ISO 파일을 선택합니다.
   d. 유형에서 USB 드라이브를 선택합니다.
   e. 드라이브의 경우, 생성한 FAT32 파티션에 해당하는 드라이브 문자를 선택합니다. 우리의 경우에는 `E:\`입니다.
   f. 확인을 선택합니다.



AWS DeepRacer 업데이트 파일을 NTFS 파티션으로 추출하려면

1. 사전 요구 사항 섹션에서 다운로드한 소프트웨어 업데이트 패키지의 압축을 풉니다.

> **Tip**
>
> 즐겨 사용하는 도구로 파일의 압축을 풀 수 없는 경우 PowerShell Expand-Archive 명령을 사용하십시오.

2. 업데이트 패키지의 내용을 USB 드라이브의 데이터(NTFS) 파티션 루트에 추출합니다.

macOS

여기의 지침에 따라 Mac을 사용하여 AWS DeepRacer 디바이스용 업데이트 미디어를 준비하십시오.

USB 드라이브를 지우고 파티션을 나누려면

1. Mac에 USB 드라이브를 꽂습니다.

2. Command+스페이스바를 눌러 Spotlight 검색 필드를 연 다음 입력하십시오 `Disk Utility`.

   또는 Finder->응용 프로그램->유틸리티->디스크 유틸리티를 선택하여 디스크 유틸리티를 엽니다.
3. 메뉴 막대에서 보기를 선택한 다음 모든 디바이스 보기를 선택합니다.
4. 사이드바의 외장에서 포맷하려는 USB 드라이브를 선택한 다음 지우기를 선택합니다.
5. 새 창에 USB 드라이브를 지울 것인지 묻는 메시지가 표시되며 이름, 형식 및 파티션 구성표를 변경할 수 있습니다. 아직 이름을 변경할 필요는 없습니다. 형식 및 구성표에서 다음 옵션을 선택하고 지우기를 선택합니다.

   - 형식: Mac OS 확장(저널링)
   - 구성표: GUID 파티션 맵

   삭제 프로세스가 완료되면 대화상자 창에서 완료를 선택합니다.
6. 기본 디스크 유틸리티 창의 사이드바에서 USB 드라이브를 선택하고 상단의 도구 모음에서 파티션을 선택합니다. 파티션 디바이스 "***YOUR-USB-DRIVE***"?라는 제목의 팝업창이 뜹니다. 추가(+) 버튼을 선택하여 새 파티션을 생성합니다.
7. 새 파티션을 만든 후에는 파티션 정보에서 다음을 선택하고 입력합니다.

   - 이름: **BOOT**
   - 형식: MS-DOS(FAT)
   - 크기: **4** GB

   > **Tip**
   > MS-DOS(FAT)를 형식으로 선택한 후 크기 입력 상자가 회색으로 표시되면 파티션 그래프의 크기 조정 컨트롤을 **BOOT** 파티션이 4GB가 될 때까지 드래그할 수 있습니다.

   아직 적용을 선택하지 마십시오.
8. 다른 무제 파티션을 선택하고 파티션 정보에서 다음 옵션을 선택한 후 입력합니다.

   - 이름: **Data**
   - 형식: ExFAT
   - 크기: USB 드라이브의 남은 공간(GB)

적용을 선택합니다.

9. USB 드라이브에 수행될 변경 사항을 보여주는 새 창이 나타납니다. 변경 내용이 올바른지 확인하십시오. 새 파티션 생성을 확인하고 시작하려면 파티션을 선택합니다.
10. 디스크 유틸리티 콘솔의 사이드바에서 BOOT 파티션을 선택한 다음 툴바에서 정보를 선택합니다. BSD 디바이스 노드 값을 기록해 두십시오. 이 자습서에서 사용된 값과 다를 수 있습니다. 이 경우 할당된 값은 `disk4s2`입니다. USB 드라이브를 FAT32 파티션에서 부팅 가능하도록 설정할 때 이 경로를 입력해야 합니다.

USB 드라이브가 FAT32 파티션에서 부팅 가능하도록 하려면

1. 필수 조건 섹션에서 사용자 지정된 Ubuntu ISO 이미지를 다운로드했는지 확인하십시오.
2. UNetbootin을 다운로드한 후 컨텍스트 메뉴(마우스 오른쪽 버튼 클릭) 에서 열기를 선택합니다. 응용 프로그램을 열 것인지 묻는 보안 메시지가 나타나면 열기를 선택하여 UNetBootin 콘솔을 시작합니다.

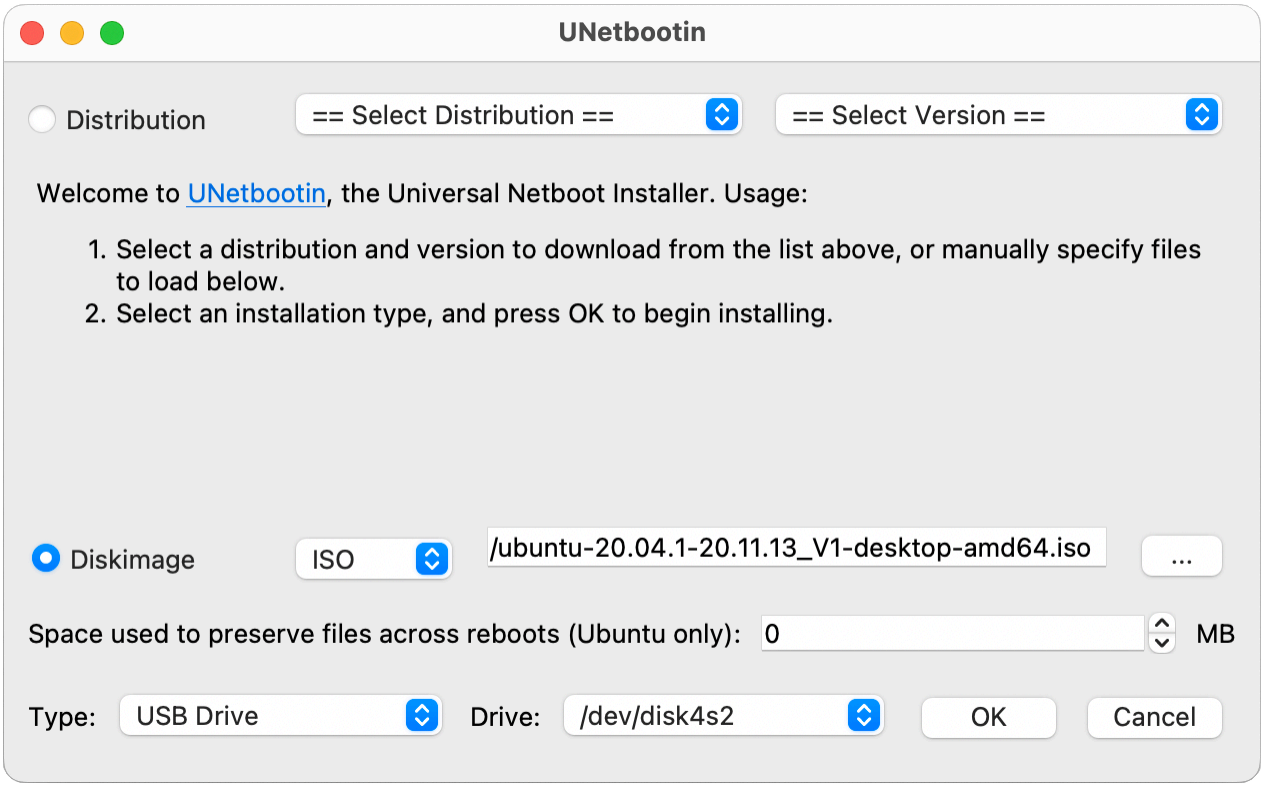
   Apple Silicon이 설치된 Mac을 사용 중이고 열기를 선택한 후 UNetBootin 콘솔이 표시되지 않는 경우, 다음 단계에 따라 Rosetta 2가 설치되어 있는지 확인하십시오.

   a. Finder > 응용 프로그램 > 유틸리티 > 터미널을 선택하여 터미널 창을 엽니다.
   b. 다음 명령을 입력하여 Rosetta 2를 설치합니다.

   ```
   softwareupdate --install-rosetta
   ```

   c. UnetBootin을 다시 열어 보십시오.
3. UNetbootin 콘솔에서 다음을 수행합니다.

   a. Disk image 라디오 버튼을 선택합니다.
   b. 디스크 이미지의 경우 드롭다운 목록에서 ISO를 선택합니다.
   c. File Picker를 연 다음 사용자 지정 Ubuntu ISO 파일을 선택합니다.
   d. 유형에서 USB 드라이브를 선택합니다.
   e. 드라이브의 경우, BOOT 파티션의 BSD 디바이스 노드를 선택하십시오(여기서는, `/dev/disk4s2`).
   f. 확인을 선택합니다.



> **Tip**
>
> /dev/disk4s2이 탑재되지 않음 알림 메시지가 표시되면 확인를 선택하여 메세지를 닫고, USB 드라이브를 뺐다가 다시 꽂은 다음 위의 단계에 따라 Ubuntu ISO 이미지를 생성합니다.

AWS DeepRacer 업데이트 파일을 ExFAT 파티션으로 추출하려면

1. 사전 요구 사항 섹션에서 다운로드한 소프트웨어 업데이트 패키지의 압축을 풉니다.
2. 업데이트 패키지의 내용을 USB 드라이브의 데이터(ExFAT) 파티션 루트에 추출합니다.

AWS DeepRacer 디바이스를 Ubuntu 20.04 소프트웨어 스택으로 업데이트하십시오.

이전 단계에서 설명한 대로 USB 업데이트 미디어를 생성하면 AWS DeepRacer 디바이스를 Ubuntu 20.04 Focal Fossa, Intel® OpenVINO™ 툴킷 2021.1.110, ROS2 Foxy Fitzroy 및 Python 3.8 등의 최신 소프트웨어 스택으로 업데이트할 수 있습니다.

> **Important**
>
> 새 AWS DeepRacer 소프트웨어 스택으로 업데이트하면 AWS DeepRacer 디바이스의 모든 데이터가 삭제됩니다.

AWS DeepRacer 디바이스 소프트웨어를 Ubuntu 20.04 스택으로 업데이트하려면

1. AWS DeepRacer 디바이스를 모니터에 연결합니다. HDMI-HDMI 케이블, HDMI-DVI 케이블 또는 이와 유사한 케이블이 필요합니다. 케이블의 HDMI 끝을 컴퓨팅 모듈의 HDMI 포트에 삽입하고 반대쪽 끝을 모니터의 호환 포트에 연결합니다.
2. USB 키보드와 마우스를 연결합니다. AWS DeepRacer 디바이스의 컴퓨팅 모듈에는 차량 전면 양쪽에 카메라가 연결된 포트를 포함하여 3개의 USB 포트가 있습니다. 4번째 USB 포트는 차량 뒷면의 컴퓨팅 배터리와 LED 테일라이트 사이의 공간에 있습니다.
3. USB 업데이트 미디어를 컴퓨팅 모듈의 사용 가능한 USB 포트에 삽입합니다. 전원을 켜거나 AWS DeepRacer 디바이스를 재설정하고 ESC 키를 반복해서 눌러 BIOS로 들어갑니다.
4. BIOS 창에서 Boot From File을 선택한 다음 부팅 파티션 이름이 있는 옵션을 선택합니다. 여기서는 BOOT라고 되어 있는 옵션을 선택한 다음, <EFI>를 선택하고, <BOOT>를 선택하고, 마지막으로 BOOTx64.EFI를 선택합니다.
5. 컴퓨팅 모듈이 부팅되면 진행 상황을 표시하는 터미널 창이 데스크탑에 나타납니다. AWS DeepRacer 디바이스는 10초 후에 업데이트 프로세스를 자동으로 시작합니다. 이 단계에서는 어떠한 입력도 제공할 필요가 없습니다.

   오류가 발생하고 업데이트에 실패하면 1단계부터 절차를 다시 시작합니다. 자세한 오류 메시지를 보려면 USB 드라이브 데이터 파티션에서 생성된 `result.log` 파일을 확인합니다.
6. 업데이트가 완료될 때까지 기다립니다. 공장 초기화가 완료되면 터미널 창이 자동으로 닫힙니다.
7. 디바이스 소프트웨어가 업데이트된 후 컴퓨팅 모듈에서 USB 드라이브를 분리합니다. 이제 AWS DeepRacer 디바이스를 재부팅하거나 종료할 수 있습니다.
8. AWS DeepRacer 디바이스는 업데이트 후 기본적으로 다음과 같은 사용자 자격 증명을 사용합니다. 처음 로그인할 때 암호를 변경하라는 메시지가 표시됩니다.

   사용자: Deepracer

   암호: `deepracer`

AWS DeepRacer를 위한 물리적 트랙 빌드

이번 단원에서는 AWS DeepRacer 모델에 사용할 물리적 트랙을 빌드하는 방법에 대해서 설명합니다. AWS DeepRacer를 자율 주행하면서 물리적 환경에서 강화 학습 모델을 테스트하려면 물리적 트랙이 필요합니다. 물리적 트랙은 훈련에 사용되는 시뮬레이션 트랙과 유사하며 배포된 AWS DeepRacer 모델을 훈련할 때 사용되는 환경을 복제합니다.

최상의 경험을 위해 사전 인쇄된 트랙과 트랙 장벽을 사용하는 것이 좋습니다. 사전 인쇄된 트랙과 장벽을 사용하면 AWS DeepRacer 트랙 환경을 원활하게 설정하고 설치할 수 있습니다. 트랙을 처음부터 새로 만드는 대신 미리 인쇄된 트랙 및 트랙 장벽 섹션을 조립합니다. 이벤트가 끝나면 사전 인쇄된 트랙과 장벽을 분해 및 보관하여 향후 이벤트에 다시 사용할 수 있습니다. 사전 인쇄된 트랙과 장벽, 이벤트 공간 및 기타 요구 사항 추정에 대한 세부 정보는 AWS DeepRacer Storefront에서 확인할 수 있습니다.

주제

- 트랙 재료와 빌드 도구
- AWS DeepRacer을 위한 트랙 설치
- AWS DeepRacer 트랙 디자인 템플릿

트랙 재료와 빌드 도구

트랙을 빌드하려면 먼저 다음 재료와 도구를 준비해야 합니다.

주제

- 필요할 수 있는 재료
- 필요할 수 있는 도구

필요할 수 있는 재료

트랙 빌드를 위해 필요한 재료는 다음과 같습니다.

- 트랙 경계 재료:

  너비가 약 2인치인 테이프를 사용해 트랙을 만들 수 있으며, 이때 테이프 색상은 어두운 색상의 트랙 표면과 대비될 수 있도록 화이트 또는 오프-화이트이어야 합니다. 표면이 어두우면 화이트 또는 오

프-화이트 테이프를 사용하십시오. 예를 들어 너비가 1.88인치인 펄 화이트 덕트 테이프 또는 너비가 1.88인치인 (점착성이 비교적 낮은) 마스킹 테이프를 사용하면 됩니다.

- 트랙 표면 재료:

  하드우드, 카페트, 콘크리트 또는 아스팔트 펠트 같이 어두운 색상이면서 단단한 바닥 위에 트랙을 만들 수 있습니다. 특히 후자는 빛 반사를 최소화하여 실제 도로 표면과 거의 흡사합니다. 연결식 폼이나 고무 패드도 훌륭한 재료입니다.

필요할 수 있는 도구

다음 도구는 트랙을 설계하여 빌드하는 데 필요하거나 효과적입니다.

- 줄자와 가위

  성능이 좋은 줄자와 가위는 트랙을 빌드하는 데 반드시 필요합니다. 아직 준비하지 못했다면 줄자는 여기에서, 혹은 가위는 여기에서 주문할 수 있습니다.
- 선택 사항인 설계 도구

  나만의 트랙을 설계하고 싶다면 각도기, 자, 연필, 칼, 나침반이 필요할 수 있습니다.

AWS DeepRacer을 위한 트랙 설치

트랙을 빌드할 때는 직선 트랙이나 단일 회전 트랙 같이 간단한 설계부터 시작하는 것이 좋습니다. 그런 다음 순환 트랙으로 넘어갈 수 있습니다. 여기에서는 단일 회전 트랙을 예로 들어 트랙을 빌드하는 각 단계에 대해서 설명합니다. 먼저 트랙의 크기 요건부터 살펴보겠습니다.

주제

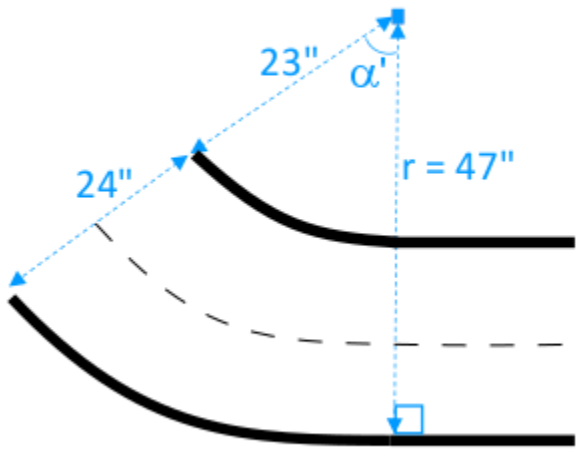
- 크기 요건
- 모델 성능 고려 사항
- 트랙 빌드 단계

크기 요건

트랙은 다음과 같은 요건을 따른다면 어떤 형상으로든 빌드할 수 있습니다.

- 최소 회전 반경:

곡선 구간에서 회전 반경(r)은 아래 그림과 같이 원심부터 바깥쪽 경계까지입니다.



최소 회전 반경(r_{min})은 코너에서 트랙 회전 각도(α)에 따라 달라지며, 다음과 같은 제한 조건을 따라야 합니다.

- 트랙의 회전 각도가 $\alpha \leq 90$ degrees이고,

  $r_{min} \geq 25$ inches

  30인치를 권장합니다.
- 트랙의 회전 각도가 $\alpha > 90$ degrees이고, α

  $r_{min} \geq 30$ inches.

  35인치를 권장합니다.

- 트랙 너비:

  트랙 너비(w_{track})는 다음 제한 조건을 따라야 합니다.

  $w_{track} \geq 24 \pm 3$ inches.
- 트랙 표면:

  트랙 표면은 평활하고 전체적으로 어두운 색상이어야 합니다. 최소 둘레 영역의 크기는 30 inches x 60 inches가 되어야 합니다.

카페트가 깔려 있거나 목재 바닥도 좋습니다. 연결식 폼이나 고무 패드는 목재보다 시뮬레이션 환경에 적합하지만 필수 조건은 아닙니다. 콘크리트 바닥은 표면의 빛 반사로 인해 문제를 일으킬 수도 있습니다.

- 트랙 장벽

  필수 조건은 아니지만 모든 지점에서 2피트 떨어진 거리에 높이가 2.5피트 이상인 균일한 색상의 장벽으로 트랙을 둘러싸는 것이 좋습니다.

모델 성능 고려 사항

트랙 빌드 방식은 훈련된 모델의 안정성과 성능에 영향을 미칠 수 있습니다. 다음은 나만의 트랙을 빌드할 때 고려해야 할 사항입니다.

1. 트랙 위에 또는 가까이 흰색 물체를 두지 마십시오. 가능하다면 트랙 또는 트랙 주변에서 흰색 물체를 제거하는 것이 좋습니다. 이는 시뮬레이션 환경에서 훈련할 때는 트랙 경계만 흰색이라는 가정을 전제로 하기 때문입니다.
2. 깨끗하고 끊김이 없는 테이프를 사용해 트랙 경계를 표시하십시오. 끊기거나 주름진 트랙 경계는 훈련된 모델의 성능에 영향을 미칠 수 있습니다.
3. 빛을 반사하는 표면은 트랙 바닥으로 사용하지 마십시오. 밝은 조명으로 인한 눈부심은 줄이는 것이 좋습니다. 직선 가장자리의 눈부심은 물체 또는 경계로 오해할 수 있습니다.
4. 트랙 라인 외에 다른 라인이 표시된 트랙 바닥은 사용하지 마십시오. 모델이 트랙 라인이 아닌 라인까지 트랙의 일부로 인식할 수 있습니다.
5. 트랙 주변에 장벽을 세워 배경 물체로 인한 주의 산만을 줄이는 것이 좋습니다.

트랙 빌드 단계

앞에서도 언급했지만 여기에서는 가장 기본적인 단일 회전 트랙을 사용합니다. 여기 지침을 수정하여 S-곡선, 순환, AWS re:invent 2018 트랙 등 더욱 복잡한 트랙을 만들 수도 있습니다.

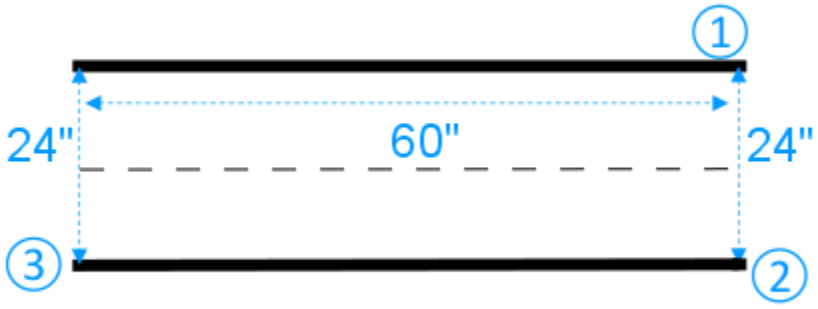
AWS DeepRacer 단일 회전 트랙을 빌드하는 방법

1. 트랙에서 직선 구간을 만들 때는 아래 단계를 따르고 다이어그램을 참조하십시오.

   a. 길이가 60인치인 테이프를 바닥에 붙여 직선 구간의 첫 번째 경계(1)를 설치합니다.

b. 줄자를 사용해 두 번째 경계의 두 끝점인 (2) 및 (3)의 위치를 찾습니다. 첫 번째 경계의 두 끝점에서 24인치를 떨어뜨려 두 끝점의 위치를 결정합니다.

c. 길이가 60인치인 테이프를 하나 더 바닥에 붙여 두 번째 경계를 설치하면서 두 끝점인 (2) 및 (3)를 연결합니다.

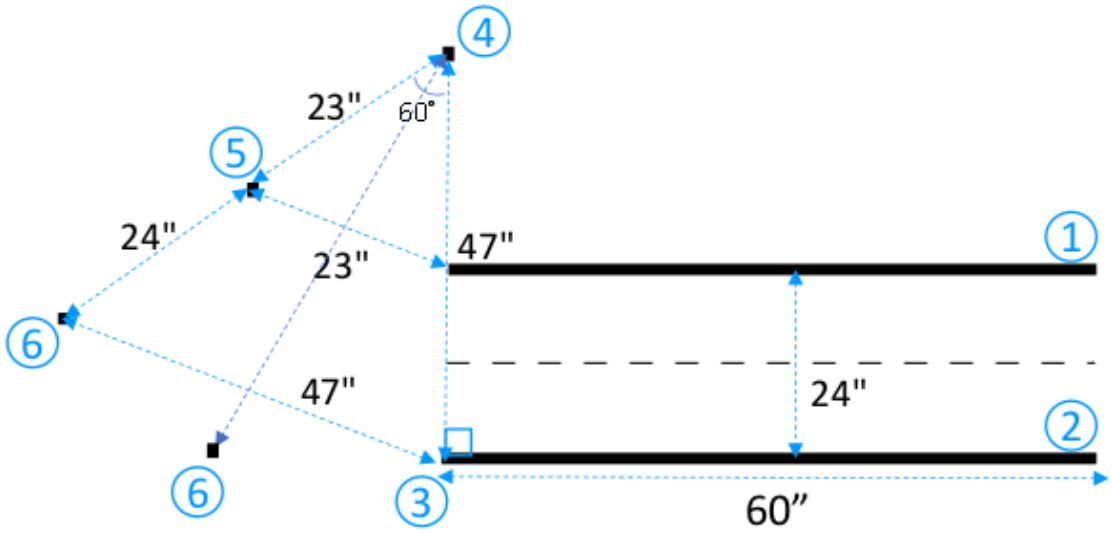


여기에서 직선 구간은 길이가 60인치이고, 너비가 24인치라고 가정합니다. 하지만 길이와 너비는 크기 요건만 충족한다면 사용 공간에 따라 조정할 수 있습니다.

2. 트랙에 60도 회전 구간을 만든다고 가정할 때 아래 단계를 따르고 다이어그램을 참조하십시오.

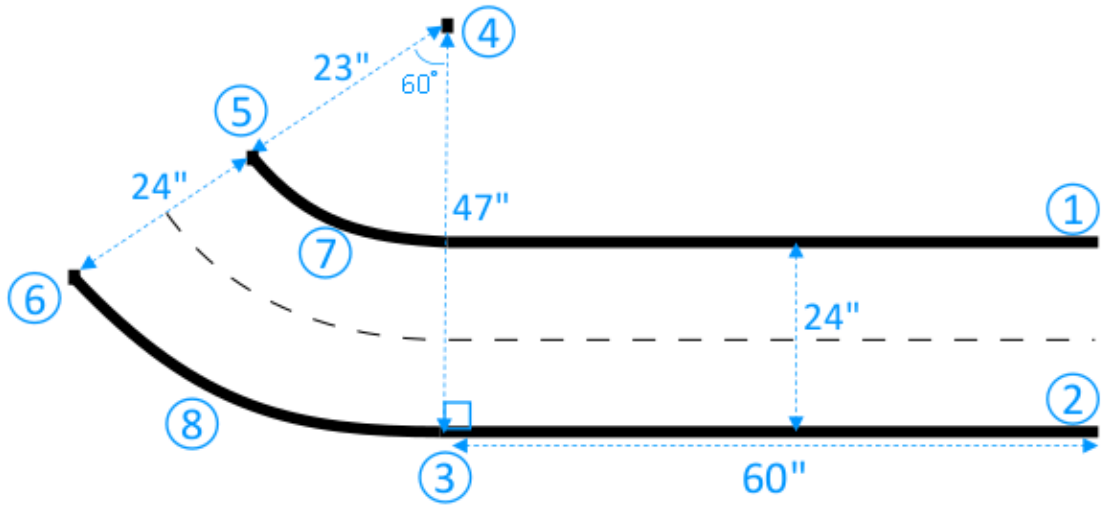
a. 줄자를 사용해 회전 반경(4-3 또는 4-6)의 원심(4)을 찾습니다. 테이프 조각을 붙여 원심을 표시합니다.

b. 정삼각형을 그립니다. 이때 세 변은 (3-4), (4-6) 및 (6-3)입니다.



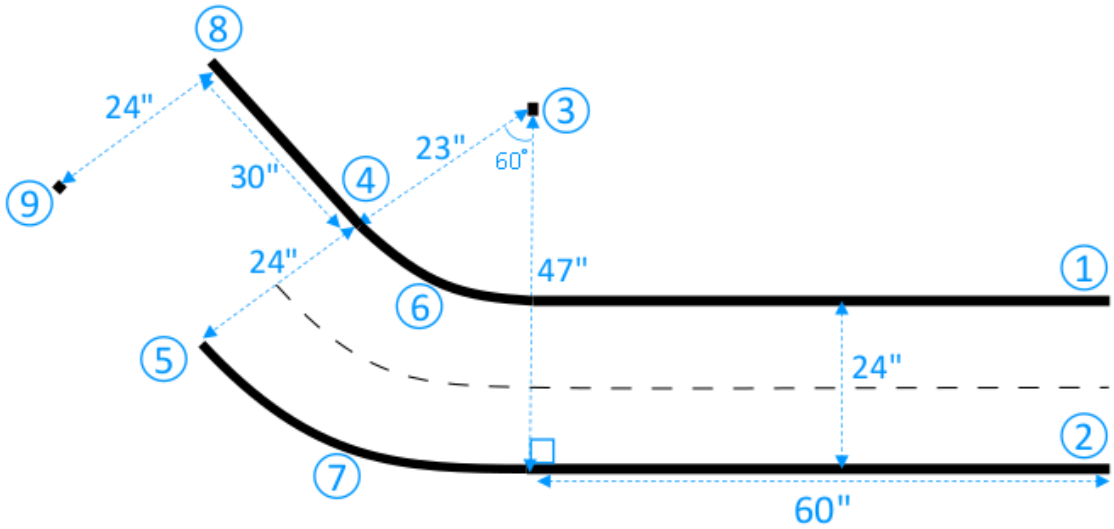
트랙을 따라 60도 회전 구간을 만들려면 정삼각형(3-4-6)을 사용해 곡선 구간의 마지막 두 끝점 (5) 및 (6)의 위치를 결정합니다. 다른 각도의 회전 구간을 원한다면 각도기(또는 각도기 앱)을 사용해 곡선 구간의 마지막 두 끝점 (5) 및 (6)의 위치를 결정할 수 있습니다. 회전 반경 편차는 단계 2에서 최소 회전 반경 요건을 충족하는 범위 내에서 허용됩니다.

c. 작은 길이의 테이프(예: 각 4인치)를 바닥에 붙이면서 곡선 경계 구간 (7)및 (8)을 설치하여 직선 경계와 연결합니다. 두 곡선 경계가 평행할 필요는 없습니다.



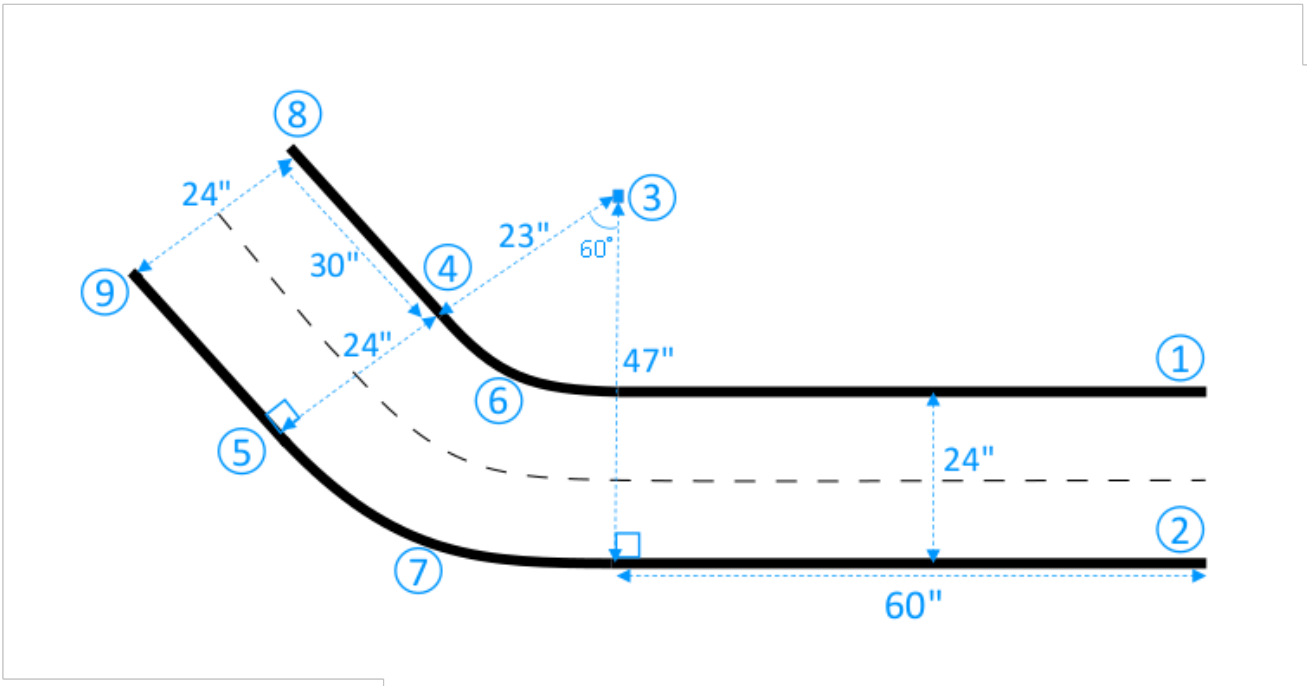
3. 다음과 같이 길이가 30인치이고 너비가 24인치인 다음 구간 길이로 트랙을 연장합니다.

   a. 길이가 30인치인 테이프를 바닥에 붙여 가장자리(3-5)에 직각인 첫 번째 경계(4-8)를 설치합니다.



b. 줄자를 사용해 두 번째 경계의 끝점(9) 위치를 결정합니다. 이때 직선 길이는 사용 공간에 맞춰 조정할 수 있습니다.

c. 길이가 30인치인 테이프를 하나 더 바닥에 붙여 가장자리(3-5)에 직각인 두 번째 경계(5-9)를 설치합니다.



여기에서 두 번째 직선 구간은 길이가 30인치이고, 너비가 24인치라고 가정합니다. 하지만 길이와 너비는 크기 요건을 충족하고 크기가 다른 트랙 구간과 일치한다면 사용 공간에 따라 조정할 수 있습니다.

4. (선택 사항) 테이프를 각각 4인치로 절단한 후 트랙 중앙을 따라 2인치 간격으로 붙여서 중앙 점선을 설치합니다.

이제 단일 회전 트랙의 빌드를 마쳤습니다. 차량이 주행 가능한 표면과 그렇지 않은 표면을 정확하게 구분할 수 있도록 트랙 표면 색상과 충분히 대비되는 색상으로 트랙 외 표면을 도색해야 합니다. 또한 안전을 위해 모든 지점에서 2피트 떨어진 거리에 높이가 2.5피트 이상인 균일한 색상의 장벽으로 트랙을 둘러싸는 것도 좋습니다.

위의 지침을 적용하여 더욱 복잡한 형상으로 트랙을 연장할 수도 있습니다.

AWS DeepRacer 트랙 디자인 템플릿

다음 트랙 디자인 템플릿은 이 섹션에 제시된 지침에 따라 구축할 수 있는 AWS DeepRacer 트랙을 보여줍니다.

> **Note**
>
> 이 섹션은 사전 인쇄된 상태로 사용 가능한 트랙용 템플릿도 제공합니다. 사전 인쇄된 트랙을 조립하는 것은 자체 재료로 트랙을 만드는 것보다 시간이 덜 걸리고 프로세스가 더 간단합니다. 사전 인쇄된 트랙과 장벽을 사용하는 것이 좋습니다. 사전 인쇄된 트랙을 구매하려면 AWS DeepRacer 스토어를 참조하십시오.

모든 트랙에 대해, 동일한 색상으로 재현하려면 다음 색상 사양을 사용하십시오.

- 녹색: PMS 3395C
- 주황색: PMS 137C
- 검은색: PMS 432C
- 흰색: CMYK 0-0-2-9

이 트랙은 표면에 다음과 같은 재료를 사용하여 테스트되었습니다.

- 비닐

  트랙이 무게가 13온스이고, 눈부심을 줄이기 위해 무광 처리된 스크림 비닐에 인쇄되었습니다. 비닐은 일반적으로 카페트보다 저렴하면서 성능도 우수합니다. 내구성은 카페트보다 떨어집니다.
- 카페트

  트랙은 무게가 8온스고, 염료 승화 방식으로 만들어졌고, 전면이 폴리에스터 재질이고 후면이 라텍스 고무 재질인 카페트에 인쇄되었습니다. 카페트는 내구성이 좋아 성능이 우수하지만 고가입니다.

트랙은 넓은 크기 때문에 재료 조각 하나에 모두 인쇄하기 어렵습니다. 따라서 각 재료 조각을 연결할 때 트랙 라인을 정확하게 정렬해야 합니다.

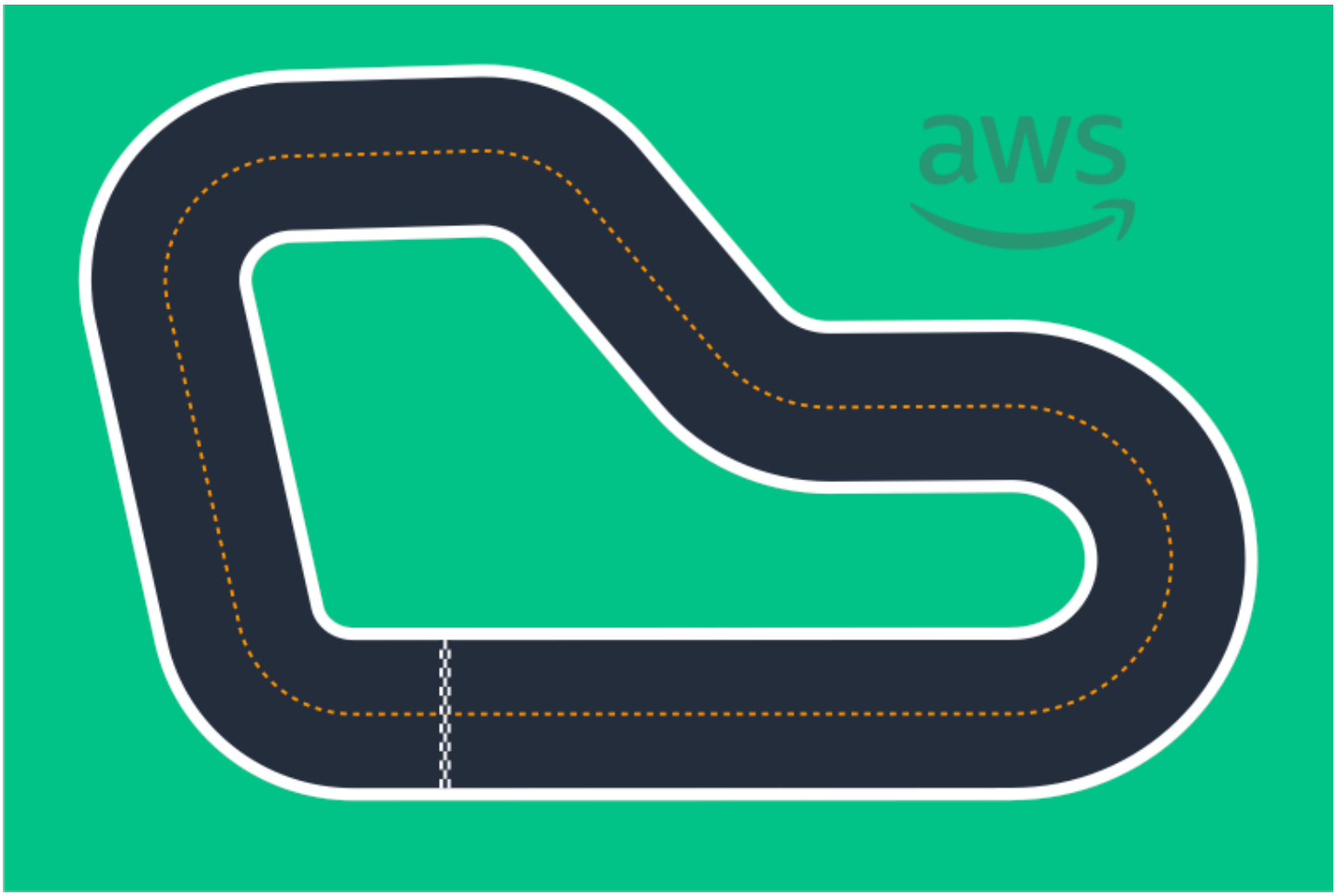
주제

- AWS DeepRacer A~Z 스피드웨이(기본) 트랙 템플릿
- AWS DeepRacer 스마일 스피드웨이(중급) 트랙 템플릿

- AWS DeepRacer RL 스피드웨이(고급) 트랙 템플릿
- AWS DeepRacer 단일 회전 트랙 템플릿
- AWS DeepRacer S-곡선 트랙 템플릿
- AWS DeepRacer 순환 트랙 템플릿

AWS DeepRacer A~Z 스피드웨이(기본) 트랙 템플릿

AWS DeepRacer A~Z 스피드웨이(기본) 트랙은 AWS DeepRacer 역사상 가장 인기 있는 물리적 경쟁 트랙입니다. 원래 AWS re:invent 2018에 출시되었으며 모든 AWS DeepRacer 물리 경쟁 트랙 중에서 설치 공간이 가장 작습니다. AWS DeepRacer Storefront에서 사전 인쇄된 트랙을 구매할 수 있습니다.

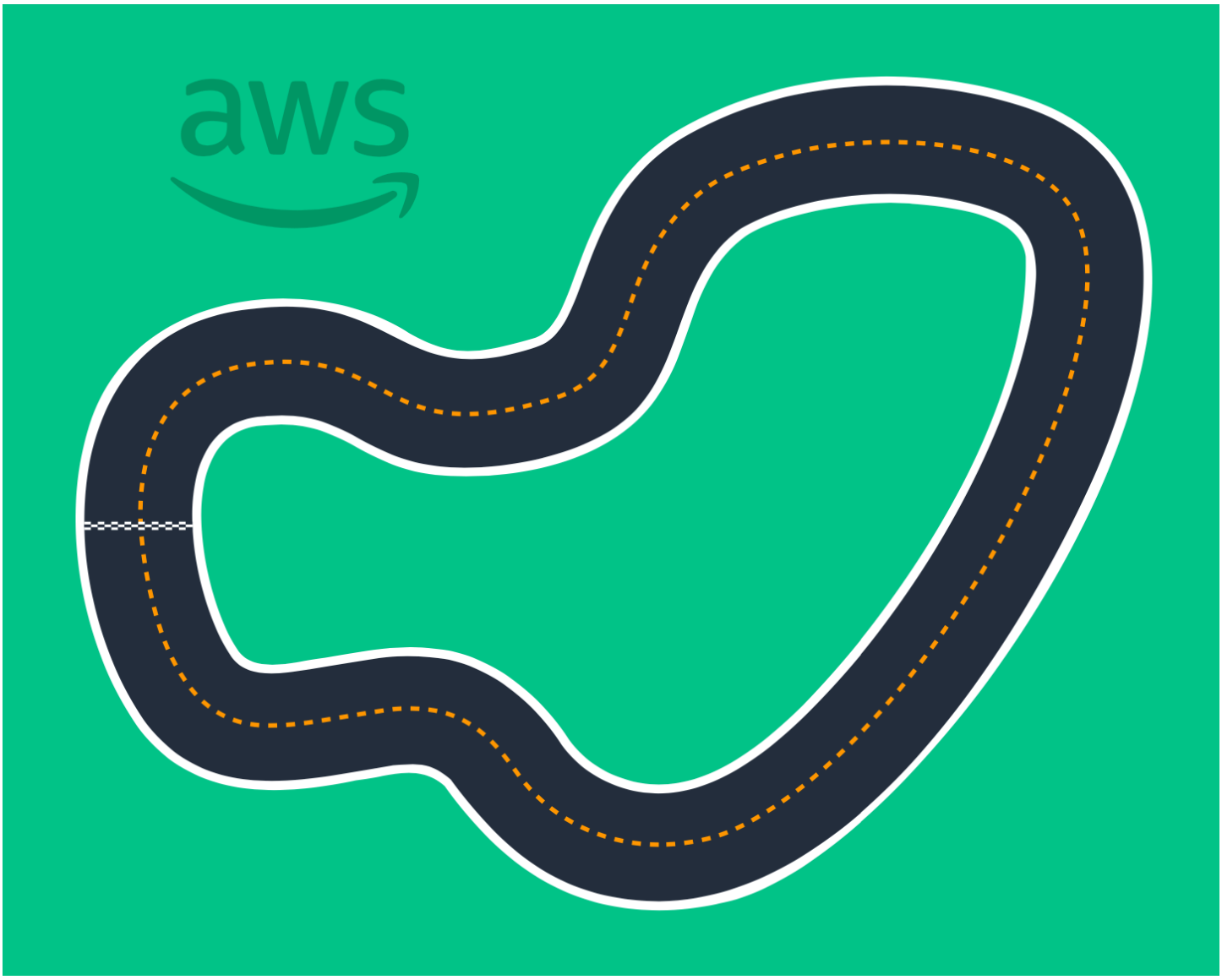


이 트랙은 초보자 이벤트와 초보 레이서에게 추천합니다. 다양한 코스와 직선로를 갖추고 있어 초보 레이서와 숙련된 레이서 모두에게 매력적인 도전을 선사합니다. AWS DeepRacer A~Z 스피드웨이(기본) 트랙은 콘솔에서 사용할 수 있는 가상 트랙을 물리적으로 1:1 로 재현한 것입니다. 이를 통해 레이서들은 가상 환경에서 모델을 학습시킨 다음 물리적 AWS DeepRacer 디바이스에 모델을 배포하여 실제 트랙에서 자율 경주를 펼칠 수 있습니다.

자신만의 A~Z 스피드웨이(기본) 트랙을 인쇄하거나 생성하려면 이 AWS DeepRacer A~Z 스피드웨이(기본) 파일을 다운로드하십시오.

AWS DeepRacer 스마일 스피드웨이(중급) 트랙 템플릿

AWS DeepRacer 스마일 스피드웨이 트랙은 원래 AWS DeepRacer 챔피언십 2019 트랙으로 출시되었습니다. AWS DeepRacer Storefront에서 사전 인쇄된 트랙을 구매할 수 있습니다.

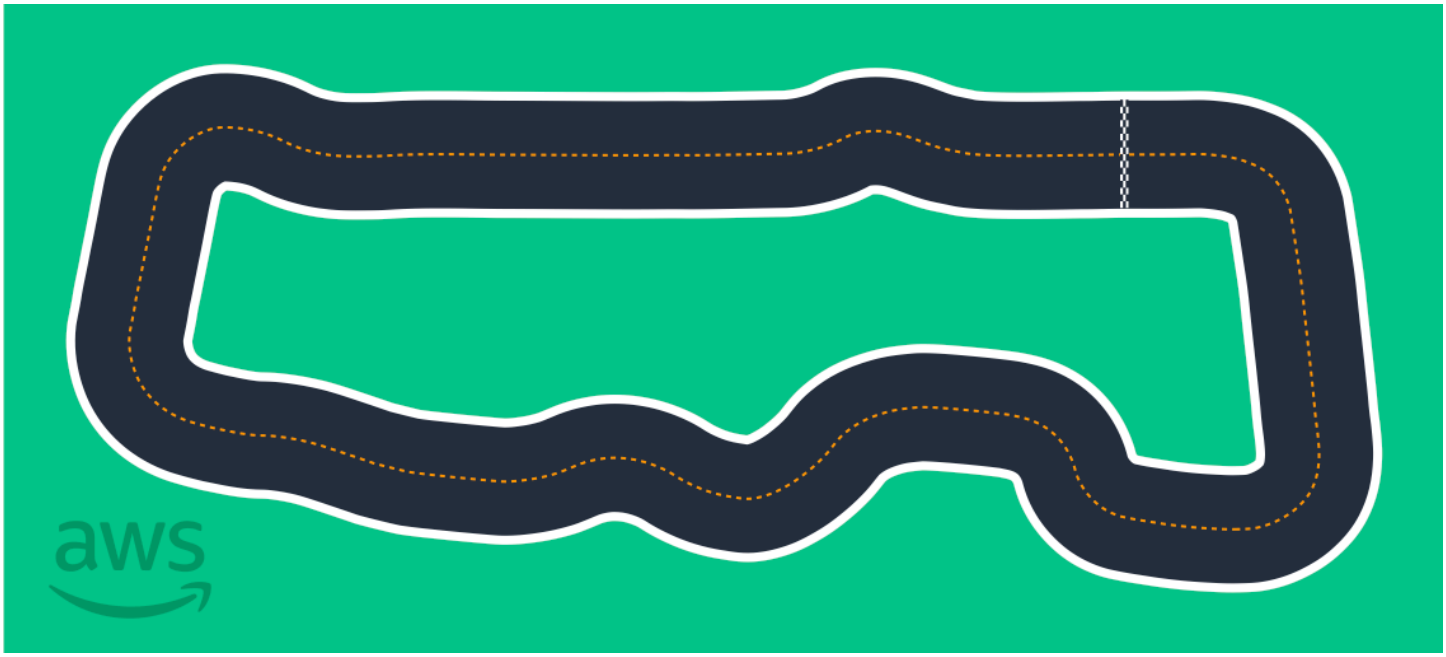


숙련된 레이서와 물리적 공간이 더 넓은 이벤트에는 이 중급 트랙을 사용하는 것이 좋습니다. 콘솔에서 사용할 수 있는 가상 트랙을 물리적으로 1:1 로 재현한 것입니다. 이를 통해 레이서들은 가상 환경에서 모델을 학습시킨 다음 물리적 AWS DeepRacer 디바이스에 모델을 배포하여 실제 트랙에서 자율 경주를 펼칠 수 있습니다.

AWS DeepRacer 스마일 스피드웨이(중급) 트랙을 인쇄하거나 생성하려면 이 AWS DeepRacer 스마일 스피드웨이(중급) 트랙 파일을 다운로드하십시오.

AWS DeepRacer RL 스피드웨이(고급) 트랙 템플릿

AWS DeepRacer RL 스피드웨이(고급) 트랙(일명 AWS DeepRacer 서밋 스피드웨이)은 원래 2022년 AWS DeepRacer 서밋을 위해 출시되었으며, AWS DeepRacer 역사상 가장 긴 물리적 트랙입니다. AWS DeepRacer Storefront에서 사전 인쇄된 트랙을 구매할 수 있습니다.

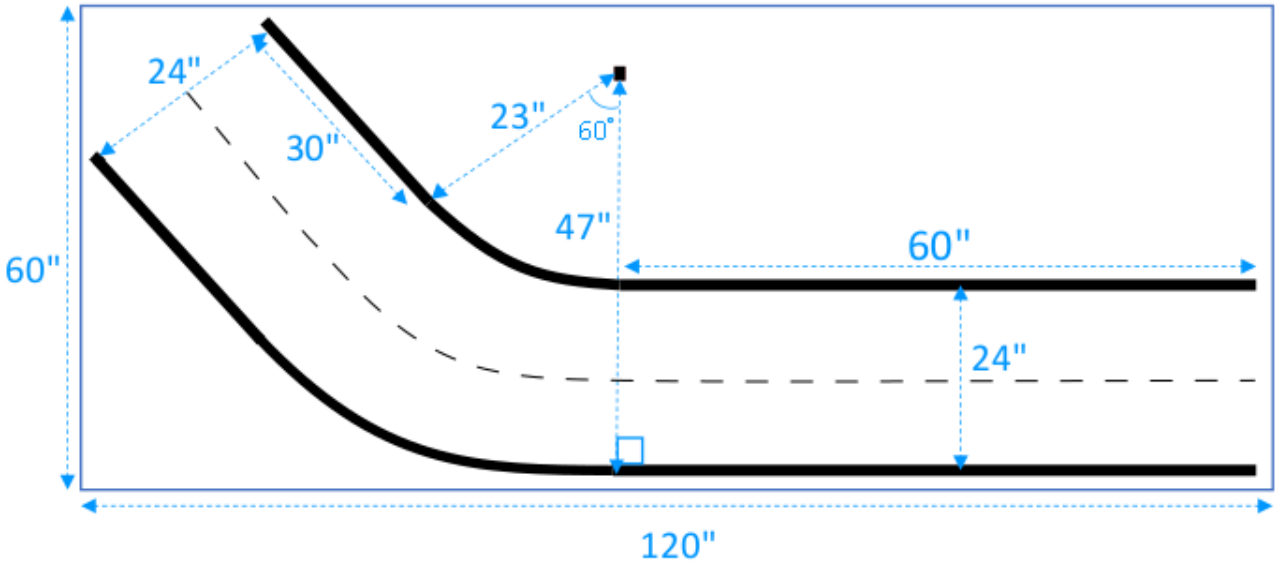


숙련된 레이서가 참가하는 이벤트에는 AWS DeepRacer RL 스피드웨이(고급) 트랙을 사용하는 것이 좋습니다. 직선 주행으로 빠르게 달리는 것을 즐기는 레이서에게는 매력적인 도전을 선사합니다. AWS DeepRacer RL 스피드웨이(고급) 트랙은 콘솔에서 사용할 수 있는 가상 트랙을 1:1 물리적으로 재현한 것입니다. 이를 통해 레이서들은 가상 환경에서 모델을 학습시킨 다음 실제 AWS DeepRacer 디바이스에 모델을 배포하여 실제 트랙에서 자율 경주를 펼칠 수 있습니다.

자체 AWS RL Speedway(고급) 트랙을 인쇄하거나 생성하려면이 AWS DeepRacer RL Speedway(고급) 트랙 파일을 다운로드합니다.

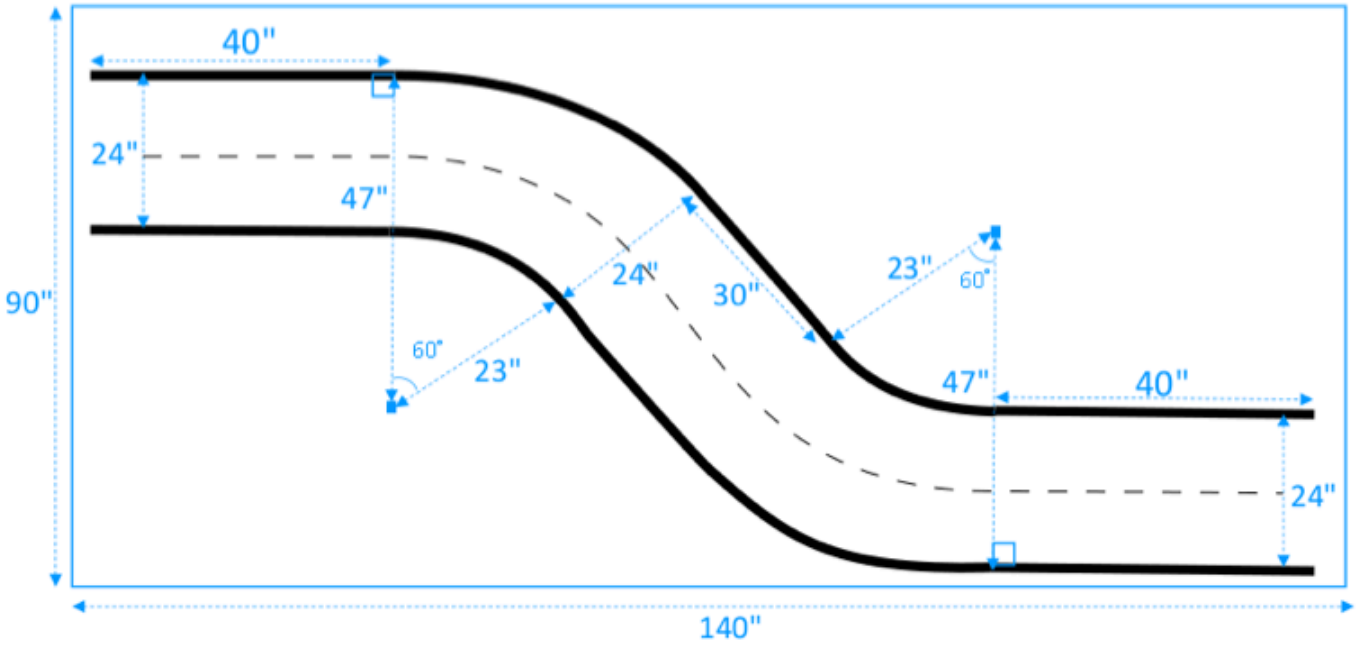
AWS DeepRacer 단일 회전 트랙 템플릿

이 기본 트랙 템플릿은 곡선 구간 1곳으로 연결된 직선 구간 2곳으로 구성됩니다. 이 트랙에서 훈련하는 모델은 AWS DeepRacer 차량이 직선으로 주행하거나, 혹은 한 방향으로 회전하도록 해야 합니다.



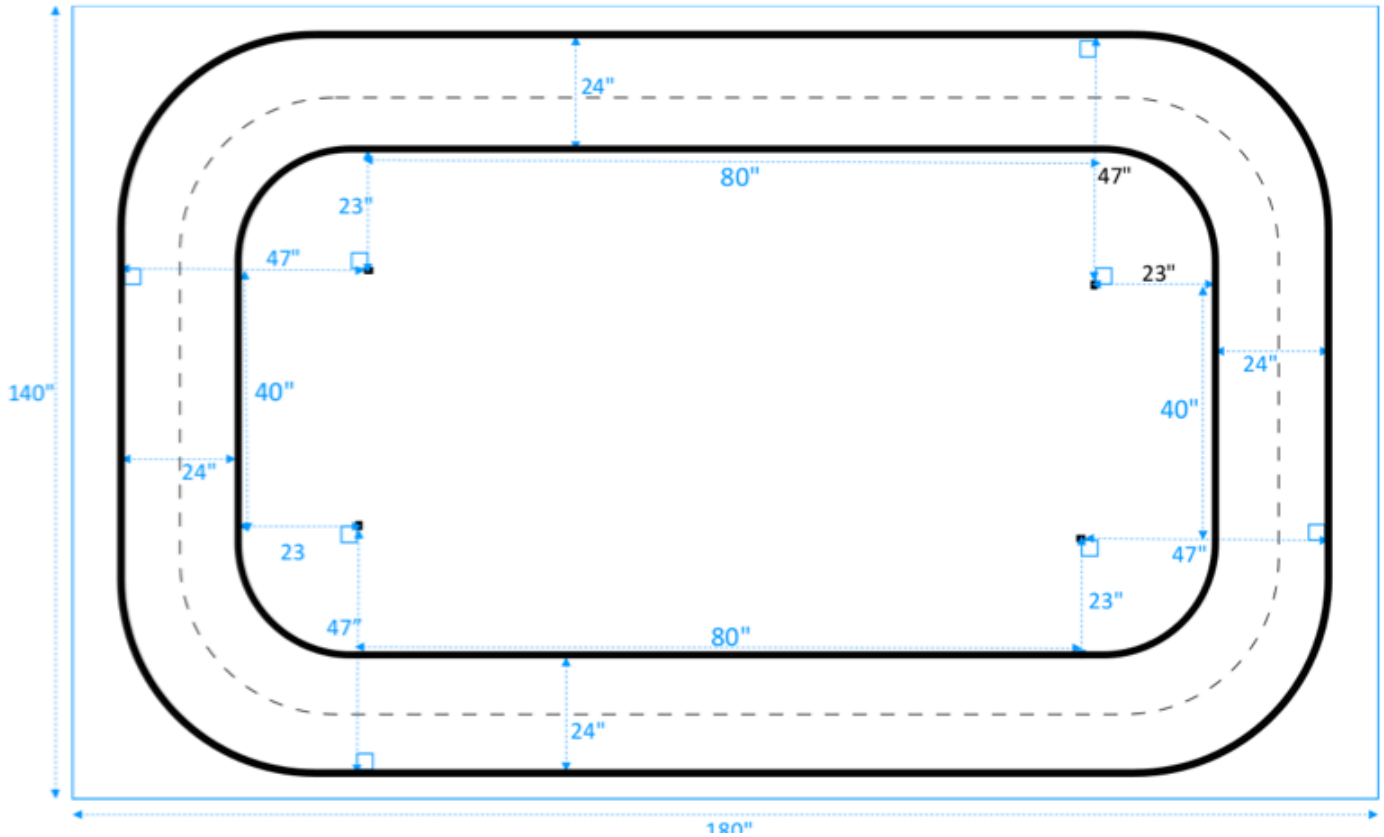
AWS DeepRacer S-곡선 트랙 템플릿

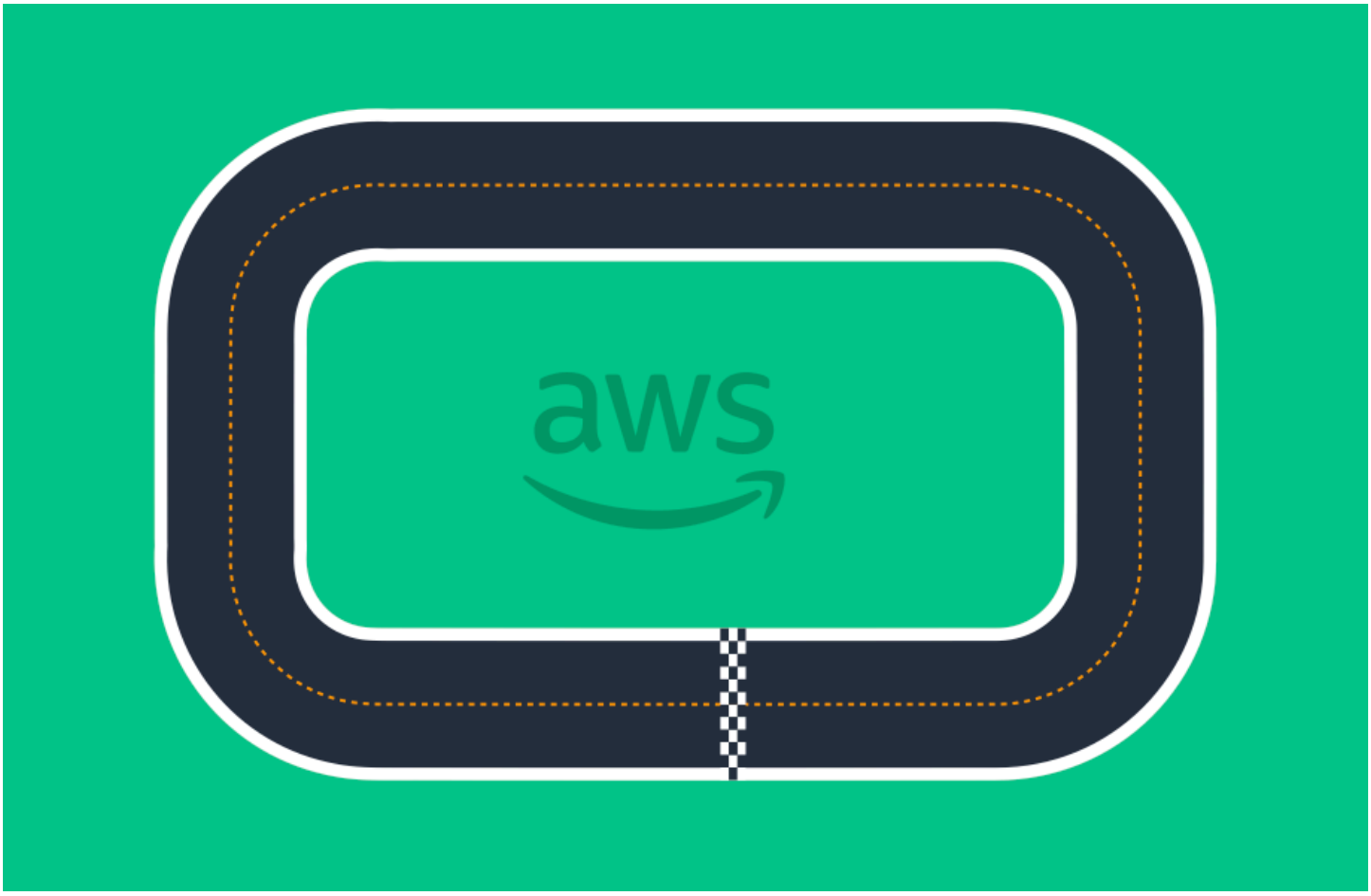
이 트랙은 단일 회전 트랙보다 더욱 복잡합니다. 모델이 두 방향으로 회전하는 방법을 학습해야 하기 때문입니다. 첫 번째 회전 이후 반대 방향으로 회전하면 단일 회전 트랙의 빌드 지침을 이 트랙으로 쉽게 연장할 수 있습니다.



AWS DeepRacer 순환 트랙 템플릿

이 규칙적인 순환 트랙은 90도로 반복되는 단일 회전 트랙입니다. 전체 트랙을 설치하려면 둘레 영역이 더욱 커야 합니다.


24"
80"
47"
23"
47"
23"
140"
40"
24"
40"
24"
23
47"
23"
47"
80"
24"
180"


aws

AWS DeepRacer 레이스에 참가하십시오

시뮬레이션에서 모듈 훈련과 평가를 성공적으로 마쳤으면 레이스에 참가하여 자신의 모델 성능을 다른 레이서의 모델과 비교할 수 있습니다. 레이싱은 모델에 대한 피드백을 받고, 보상과 상을 받고, 다른 AWS DeepRacer 커뮤니티 구성원을 만나고, 기술을 배우고 향상시킬 기회에 대해 듣고, 즐거운 시간을 보낼 수 있는 재미있는 활동입니다.

레이스는 오프라인 또는 온라인(가상)으로 진행될 수 있으며, 가상 레이스는 라이브 레이스로 동기식으로 구성되거나 클래식 레이스로 비동기식으로 구성될 수 있습니다. 라이브 및 클래식 가상 레이스는 비공개 또는 공개적으로 방송할 수 있습니다.

이 섹션에서는 AWS DeepRacer 리그 가상 서킷 레이스 또는 커뮤니티 기반 가상 레이스에 참가하는 방법과 다양한 형식 지정 옵션에 대해 설명합니다.

AWS DeepRacer 레이싱 이벤트 유형

이벤트는 스폰서 또는 주최자에 의해 분류될 수 있습니다. AWS DeepRacer 리그와 커뮤니티 레이싱 이벤트는 모두 실제 트랙에서 직접 또는 가상 트랙에서 온라인으로 이루어질 수 있습니다.

- AWS후원 레이싱 이벤트 -에서 후원하는 레이싱 이벤트를 AWS DeepRacer 리그 이벤트 AWS 라고 하며 모든 AWS DeepRacer 사용자에게 공개됩니다. 처음 레이스를 시작하는 선수는 월간 가상 레이스에 참가하여 리그 여정을 시작할 수 있습니다. 레이서가 레이스에 모델을 제출하면 포인트를 획득하고 전국 및 지역 시즌 순위를 받게 됩니다.

- 커뮤니티 후원 레이싱 이벤트 - AWS DeepRacer 사용자가 주최한 이벤트를 커뮤니티 레이싱 이벤트라고 합니다.

온라인 AWS후원 또는 커뮤니티 후원 레이스 참가

AWS DeepRacer 콘솔을 사용하여 AWS DeepRacer 리그 가상 서킷 이벤트 또는 커뮤니티 기반 온라인 레이스에 참가할 수 있습니다.

- 모든 AWS DeepRacer 사용자는 AWS DeepRacer 리그 가상 서킷에서 열리는 모든 온라인 레이스에 참가할 수 있습니다.
- 커뮤니티 레이싱 가상 이벤트는 초대된 사용자만 액세스하거나 참가할 수 있습니다. 사용자는 경주 주최자가 보내거나 다른 경주 참가자가 전달한 초대 링크를 받으면 초대됩니다.

주제

- the section called “ 가상 서킷 레이스에 참여하십시오.”
- the section called “ 커뮤니티 레이스 참가 ”
- the section called “ 라이브 레이스에 참여하십시오.”
- the section called “레이싱 이벤트 용어”

AWS DeepRacer 리그 가상 서킷 레이스에 참가하십시오

이 단원에서는 AWS DeepRacer 콘솔에서 훈련된 모델을 가상 서킷 레이스에 제출하는 방법을 알아봅니다.

AWS DeepRacer 리그 가상 서킷에 참가하려면

1. AWS DeepRacer 콘솔에 로그인합니다.
2. 기본 탐색 창에서 AWS 가상 서킷을 선택합니다.
3. AWS 가상 서킷 페이지의 오픈 레이스 섹션에서 레이스 참가를 선택합니다.
4. AWS DeepRacer 리그 레이싱 이벤트에 처음 참가하는 경우, AWS DeepRacer 리그 레이서 이름 아래 레이서 이름에 별칭을 설정하십시오.
5. 모델 선택의 모델 목록에서 사용하려는 모델을 선택합니다. 모델이 트랙 모양을 감당할 수 있도록 훈련되었는지 확인하십시오.
6. AWS DeepRacer League 이벤트에 처음 참가하는 경우 리그 요구 사항에서 거주 국가를 선택하십시오. 거주 국가를 선택하고 첫 번째 모델을 제출하면 해당 모델은 레이싱 시즌 동안 확정되며 상품 수여 시 확인됩니다. 그런 다음 체크박스를 선택하여 이용 약관에 동의하십시오.
7. 레이스 참가를 선택하여 제출을 완료하십시오. 각 레이스의 제출 할당량은 50입니다.

   모델이 제출되면 AWS DeepRacer 콘솔이 평가를 시작합니다. 이 프로세스에는 최대 10분이 걸릴 수 있습니다.
8. 레이스 페이지에서 레이스 세부 정보를 검토하십시오.
9. 레이스 페이지에서 레이서 이름의 제출 상태를 확인합니다.
10. 레이스 페이지의 리더보드에서 순위 목록을 보면서 자신의 모델과 다른 참가자 모델의 경쟁 결과를 확인할 수 있습니다.

모델이 3바퀴 연속 주행 시험을 완주하지 않으면 리더보드 아래 순위 목록에 포함되지 않습니다. 리더보드 순위에는 최고의 성과를 낸 기록이 반영됩니다. 또한 해당 국가 및 지역의 다른 레이서들 중 자신의 순위를 가늠할 수 있는 전국 및 지역 시즌 순위도 받을 수 있습니다.

모델을 제출한 후에는 보상 함수를 수정하고 모델을 반복하여 성능을 개선해 보십시오. 다른 알고리즘이나 액션 스페이스로 새 모델을 학습시킬 수도 있습니다. 배우고, 적응하고, 다시 레이스하여 보상을 받을 확률을 높이십시오.

AWS DeepRacer 커뮤니티 레이스에 참여하려면

> **Note**
>
> AWS DeepRacer 커뮤니티 레이스에 참가하려면 먼저 레이스 주최자로부터 레이스 링크를 받아야 합니다.

AWS DeepRacer 레이스에 참가하라는 초대를 받으면 해당 레이스가 라이브 레이스인지 클래식 레이스인지 알아보십시오.

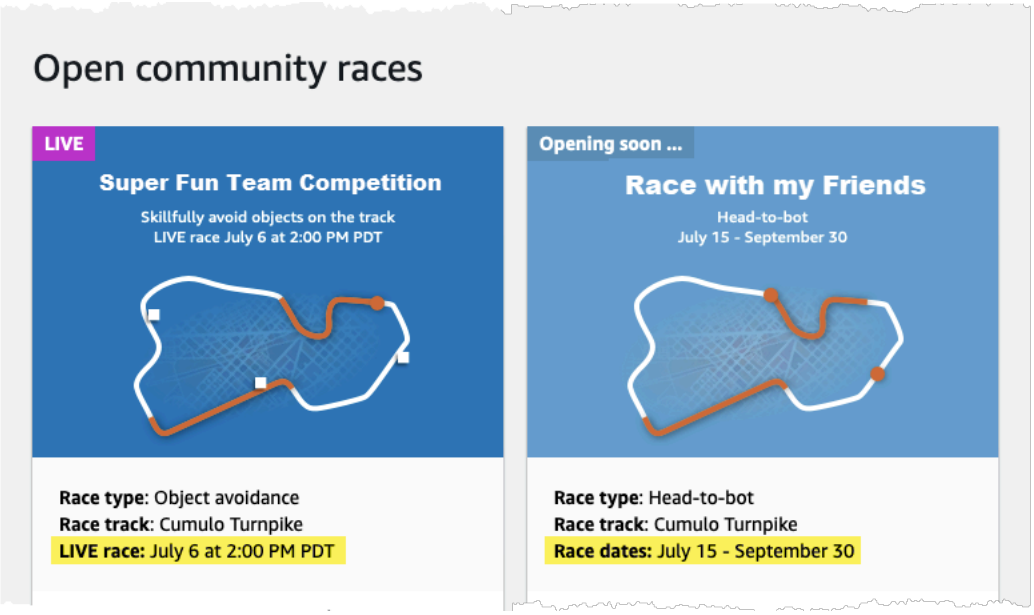
클래식 레이스

클래식 레이스는 실시간 상호 작용이 필요하지 않은 비동기 이벤트입니다. 초대 링크를 통해 레이스에 모델을 제출하고 리더보드를 볼 수 있습니다. 레이스 개막일과 마감일 내에 언제든지 모델을 무제한으로 제출하여 리더보드에서 최고의 순위를 차지할 수 있습니다. 제출된 모델의 클래식 레이스 결과 및 동영상은 레이스가 시작되자마자 리더보드 페이지에서 볼 수 있습니다. 모든 클래식 레이스는 비공개 이벤트입니다.

라이브 레이스

라이브 레이스는 다른 레이서들이 번갈아 가며 가상으로 모여 리더보드에서 가장 빠른 시간을 위해 경쟁하는 실시간 레이싱 이벤트입니다. 여러 모델이 출전할 수 있지만, 제출 창이 닫히기 전에 마지막으로 제출한 모델만 사용됩니다. 레이스 중에는 인터랙티브 속도 제어를 시도할 수 있습니다. 대화형 속도 제어는 모델의 속도 파라미터를 일시적으로 재정의하여 전략적인 실시간 조정을 할 수 있는 기회를 제공합니다. 라이브 레이스는 초대된 레이서들 사이에서 비공개로 방송하거나 누구나 볼 수 있도록 공개적으로 방송할 수 있습니다.

초대장에 대회 형식이 명시되어 있지 않은 경우 레이스 카드를 확인하십시오. 라이브 레이스에서는 “LIVE”라고 표시되며 동기 이벤트의 날짜와 시간을 알려줍니다. 클래식 레이스는 비동기 대회의 날짜 범위를 제공합니다.



레이스 참가자로 AWS DeepRacer 커뮤니티 레이스에 참가하려면

를 처음 AWS 사용하고 AWS DeepRacer 커뮤니티 레이스에 참가하라는 초대를 받는 경우 새 사용자로 가입하려면의 단계를 따르세요. 활발한 커뮤니티 레이스에 초대를 받았고 이전에 AWS DeepRacer 레이스에참가한 적이 있다면, 대회 형식에 맞춰 아래 클래식 레이스에 참가하려면 또는 라이브 레이스에 참가하려면의 단계를 따르십시오.

새 사용자로 참여하려면

를 처음 AWS 사용하고 AWS DeepRacer 커뮤니티 레이스에 참가하기 위한 초대 링크를 받는 경우 링크를 선택하여 AWS DeepRacer 콘솔로 이동한 다음 레이스에 참가하기 전에 AWS 계정에 가입합니다.

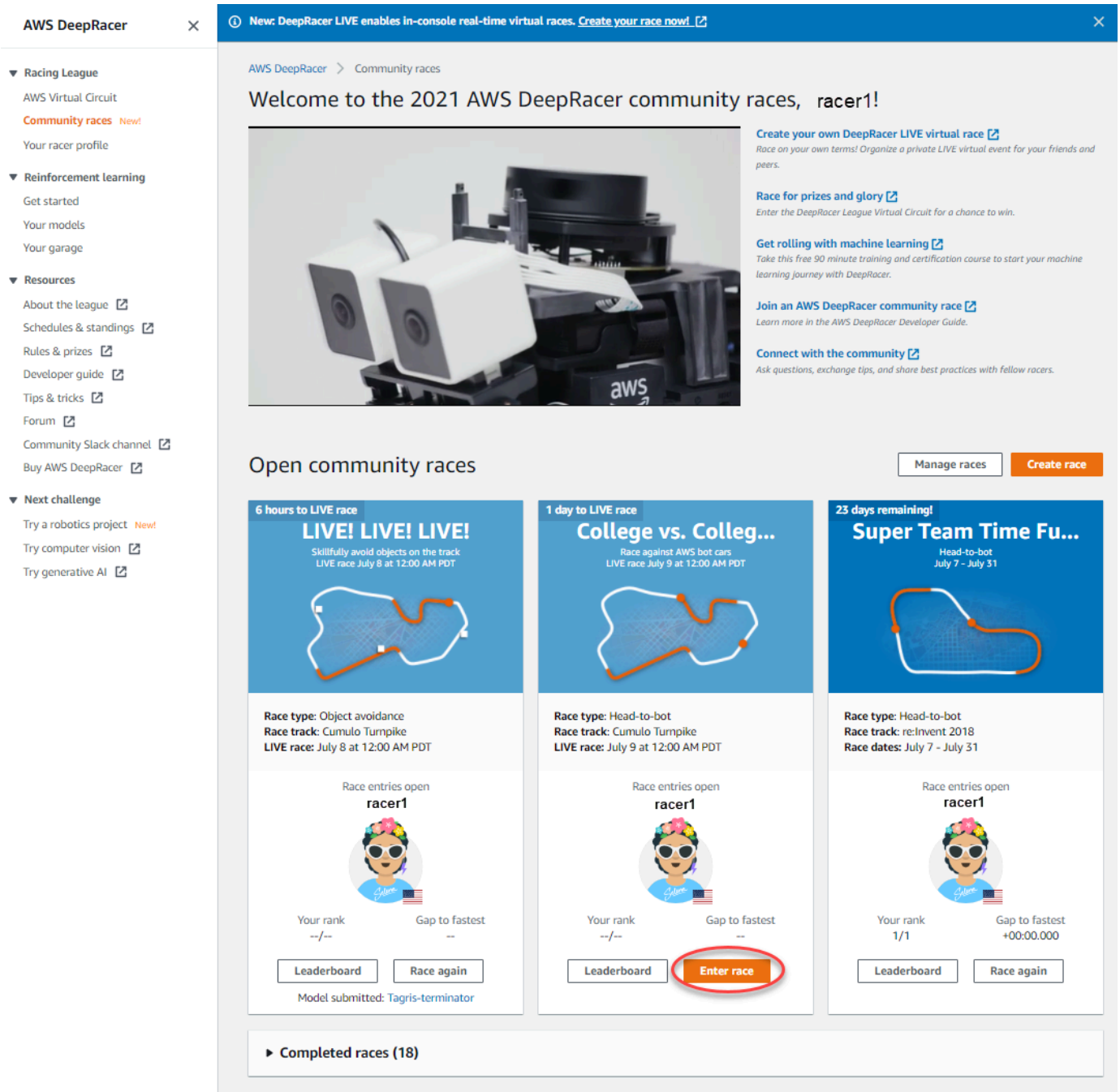
AWS DeepRacer 신규 사용자 또는 AWS DeepRacer 경주에 처음 참가하는 경우 다음 단계를 따라 AWS DeepRacer 콘솔에서 커뮤니티 레이스에 참가하십시오.

레이스에 새 사용자로 참가하려면

1. AWS DeepRacer 콘솔에서 AWS 계정을 생성합니다.
2. 설정을 마치고 로그인한 후 레이스 주최자가 공유한 링크를 선택하여 레이스 페이지를 엽니다.
3. AWS DeepRacer 레이서 이름을 생성하라는 메시지가 표시되면 모든 AWS DeepRacer 리더보드에서 ID로 사용할 이름을 입력합니다. 레이서 이름을 선택한 후에는 변경할 수 없습니다.

4. 레이스 세부 정보 페이지에서 레이싱 시작하기를 확장합니다.
5. RL 시작하기를 선택하면 자율 주행을 위한 AWS DeepRacer 모델 훈련에 대한 짧은 소개가 나옵니다.
6. AWS DeepRacer 콘솔에서 레이스에 대해 모델을 훈련하고 평가합니다.

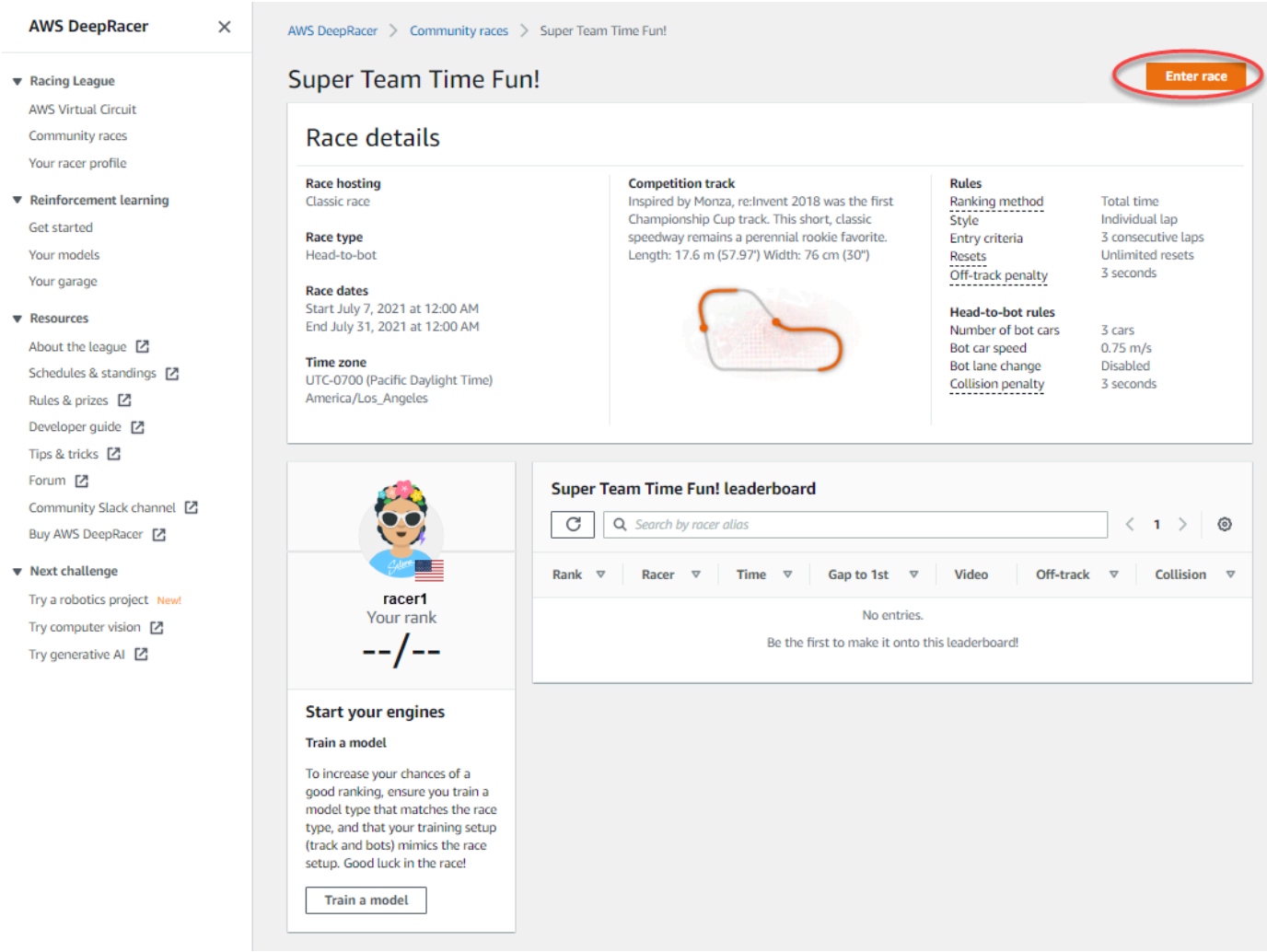
   모델 훈련에 대한 자세한 내용은 첫 번째 AWS DeepRacer 모델 훈련 단원을 참고하십시오.
7. 커뮤니티 레이스로 이동하십시오.
8. 초대받은 레이스를 찾아보십시오. 레이스 카드에서 레이스 참가를 선택합니다.



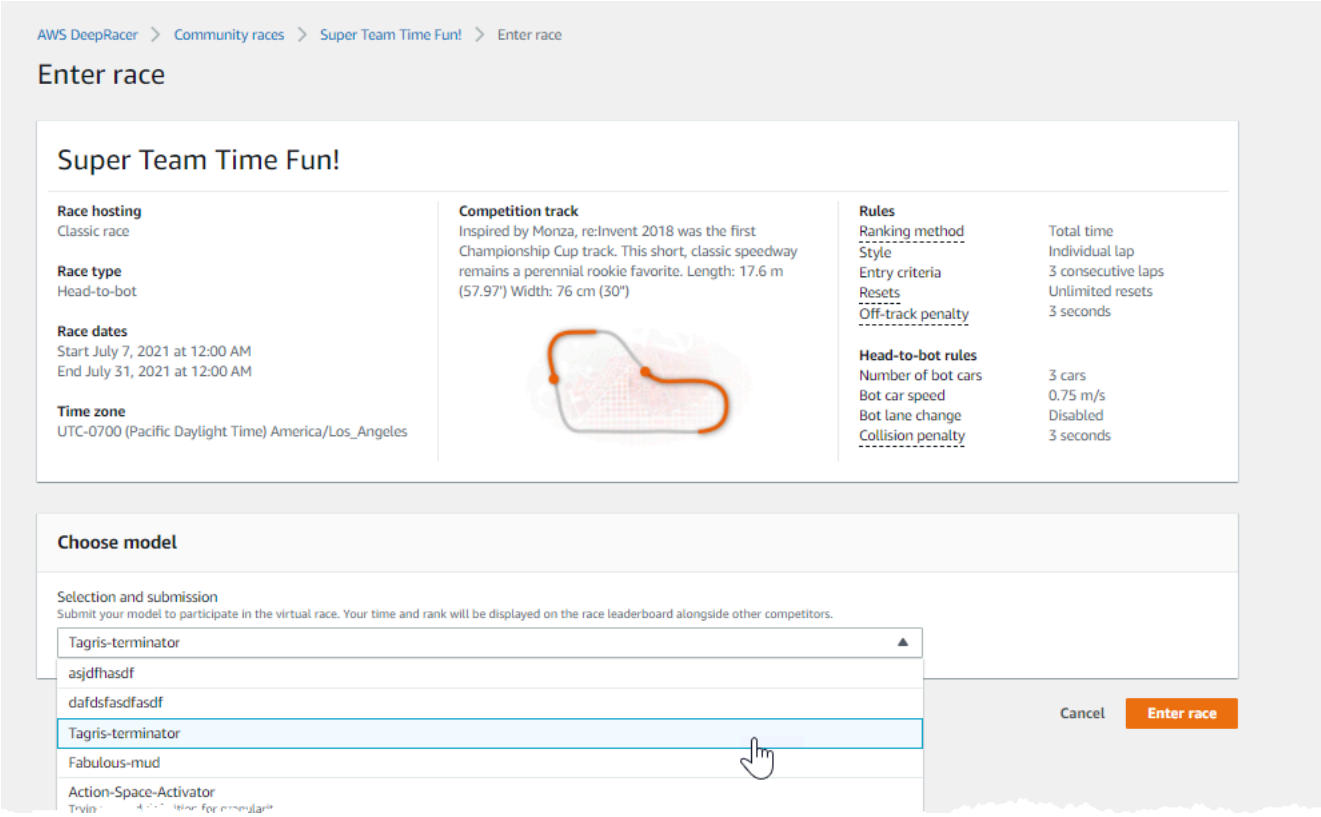
9. 레이스의 경쟁 형식에 적합한 클래식 레이스에 참가하려면 또는 라이브 레이스에 참여하려면의 단계를 따르십시오.

클래식 레이스에 참가하려면

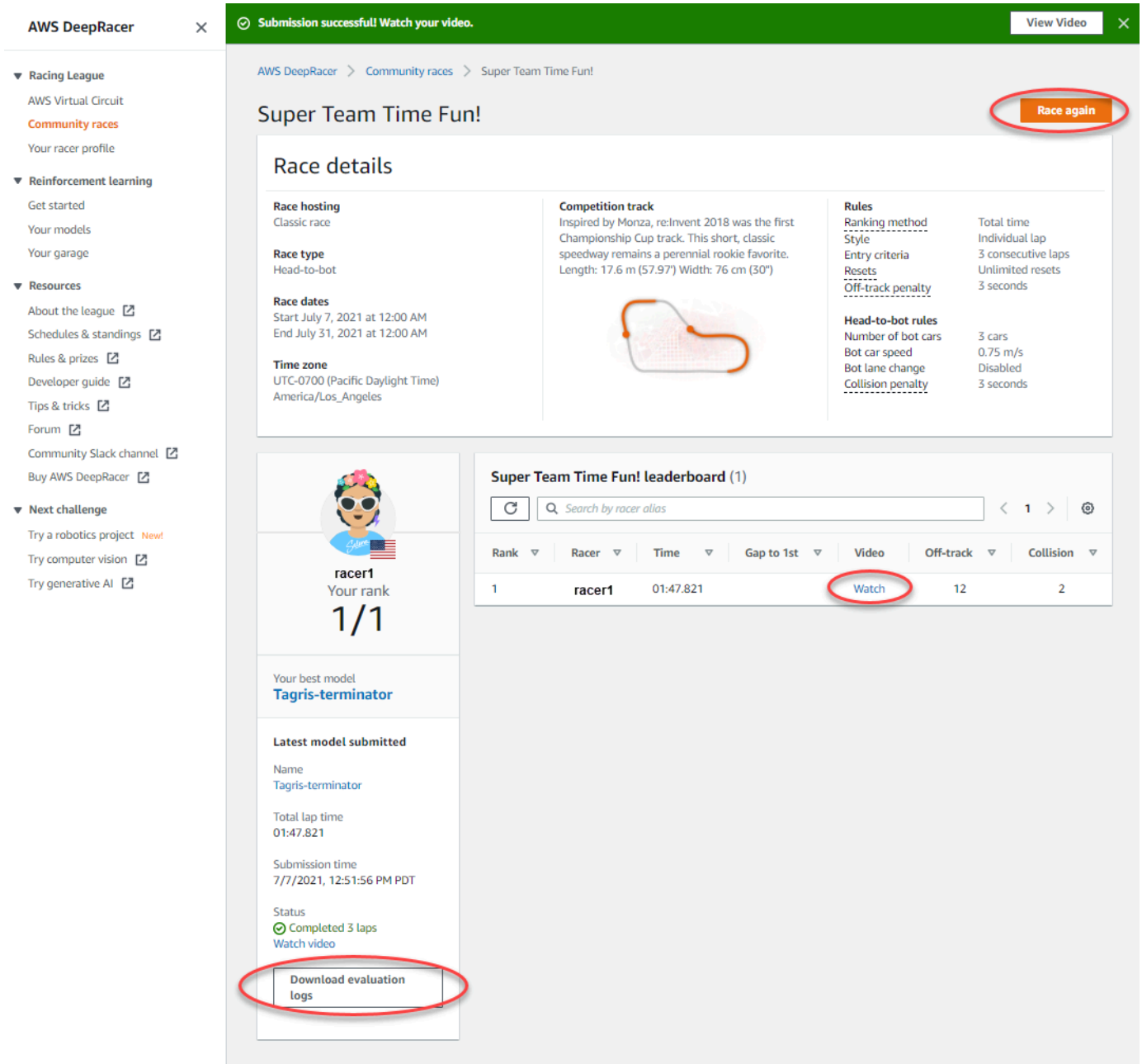
1. 레이스 주최자로부터 받은 링크를 선택합니다. AWS DeepRacer 콘솔 계정에 아직 로그인하지 않은 경우 로그인하라는 메시지가 표시됩니다.
2. AWS DeepRacer 콘솔에 로그인하고 링크를 클릭하면 레이스 페이지로 이동합니다. 레이스 페이지에는 레이스 세부 정보, 리더보드, 레이서 정보가 표시됩니다. 레이스 참가를 선택합니다.



3. 레이스 참가 페이지의 모델 선택에서 훈련된 모델을 선택한 다음 레이스 참가를 선택합니다.



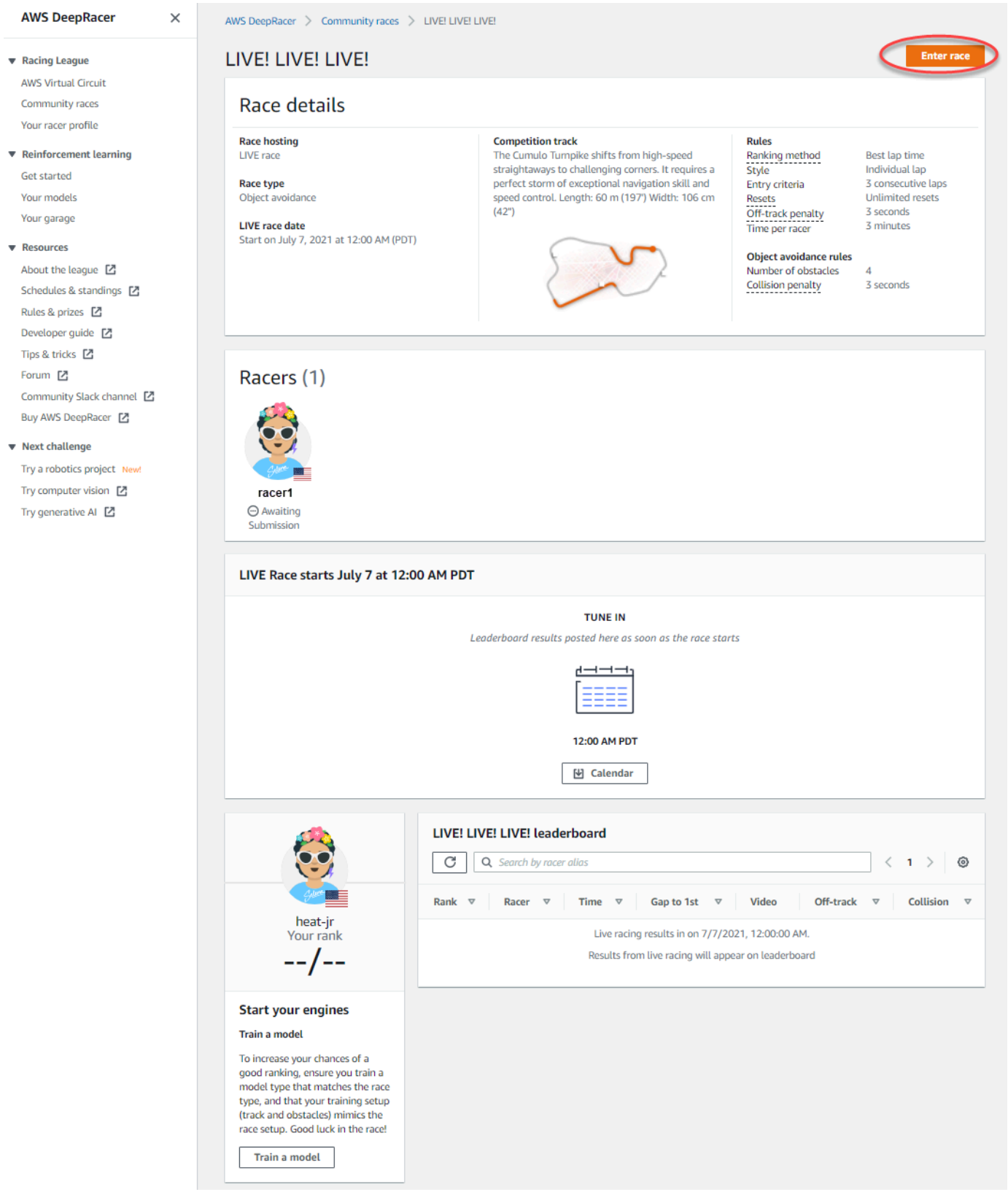
4. 모델이 레이싱 기준에 대해 성공적으로 평가되면 이벤트의 리더보드에서 다른 참가자와 비교하여 모델이 어떻게 평가되는지 확인합니다.
5. 이 외에도 보기를 선택하여 차량 성능에 대한 동영상을 보거나 평가 로그 다운로드를 선택하여 생성된 출력을 자세히 검토할 수 있습니다.



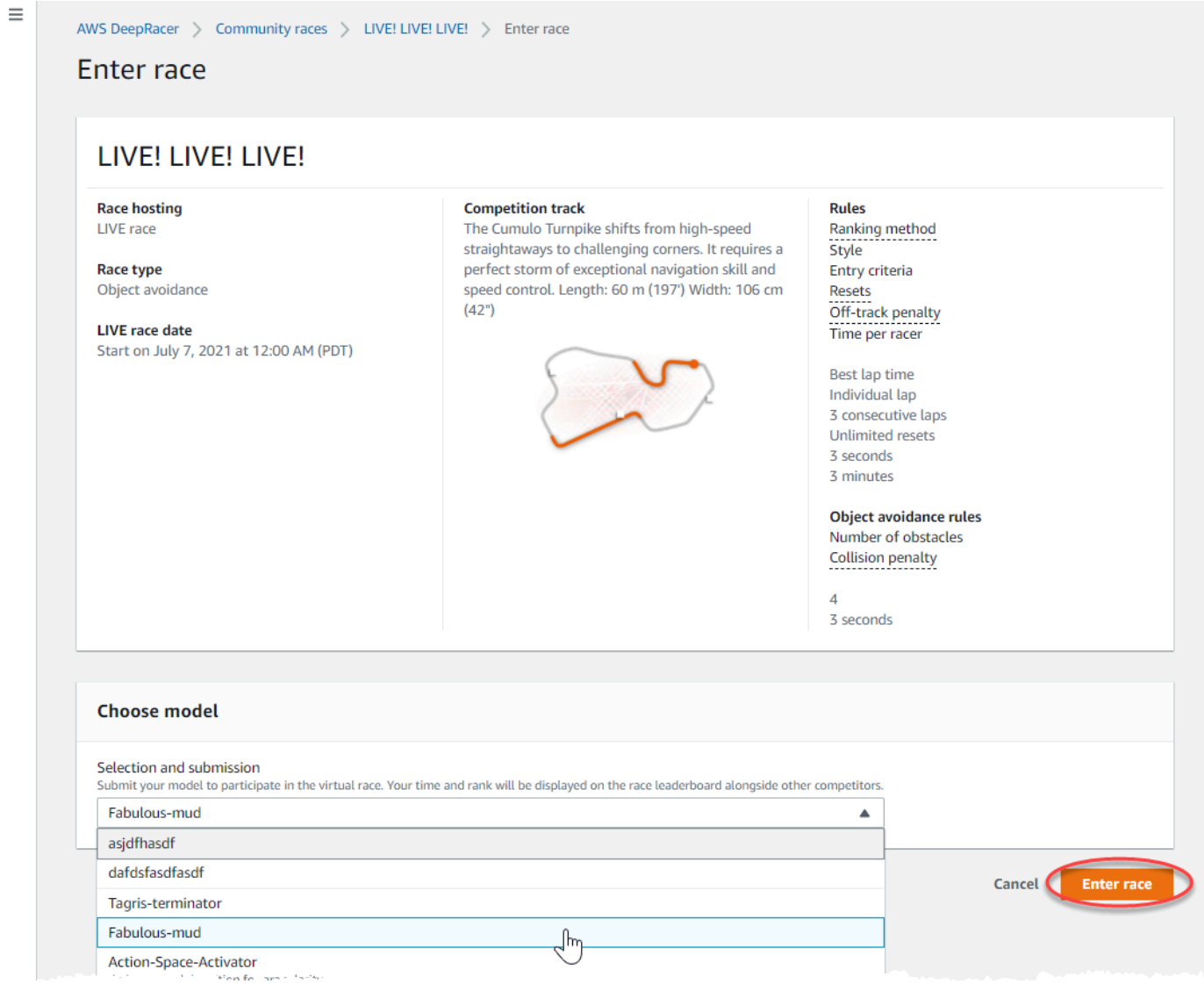
6. 레이스 다시를 선택하여 다른 모델로 참가하십시오. 레이스 개막일과 마감일 내에 언제든지 모델을 무제한으로 제출하여 리더보드에서 최고의 순위를 차지할 수 있습니다.

라이브 레이스에 참여하려면

1. 레이스 주최자로부터 받은 링크를 선택합니다. AWS DeepRacer 콘솔 계정에 아직 로그인하지 않은 경우 로그인하라는 메시지가 표시됩니다.
2. AWS DeepRacer 콘솔에 로그인하고 링크를 클릭하면 레이스 페이지로 이동합니다. 레이스 페이지에는 레이스 세부 정보와 리더보드가 표시됩니다. 레이스 참가를 선택합니다.



3. 레이스 참가 페이지의 모델 선택에서 훈련된 모델을 선택한 다음 레이스 참가를 선택합니다.



4. 모델이 레이싱 기준에 대해 성공적으로 평가되면 이벤트의 리더보드에서 다른 참가자와 비교하여 모델이 어떻게 평가되는지 확인합니다.
5. 이 외에 라이브 레이스의 경우 캘린더를 선택하여 캘린더에 라이브 레이싱 이벤트를 추가할 수도 있습니다.
6. 레이스 다시를 선택하여 다른 모델로 참가하십시오. 여러 모델이 출전할 수 있지만, 제출 창이 닫히기 전에 마지막으로 제출한 모델만 사용됩니다.

AWS DeepRacer 라이브 레이스에 참가하십시오

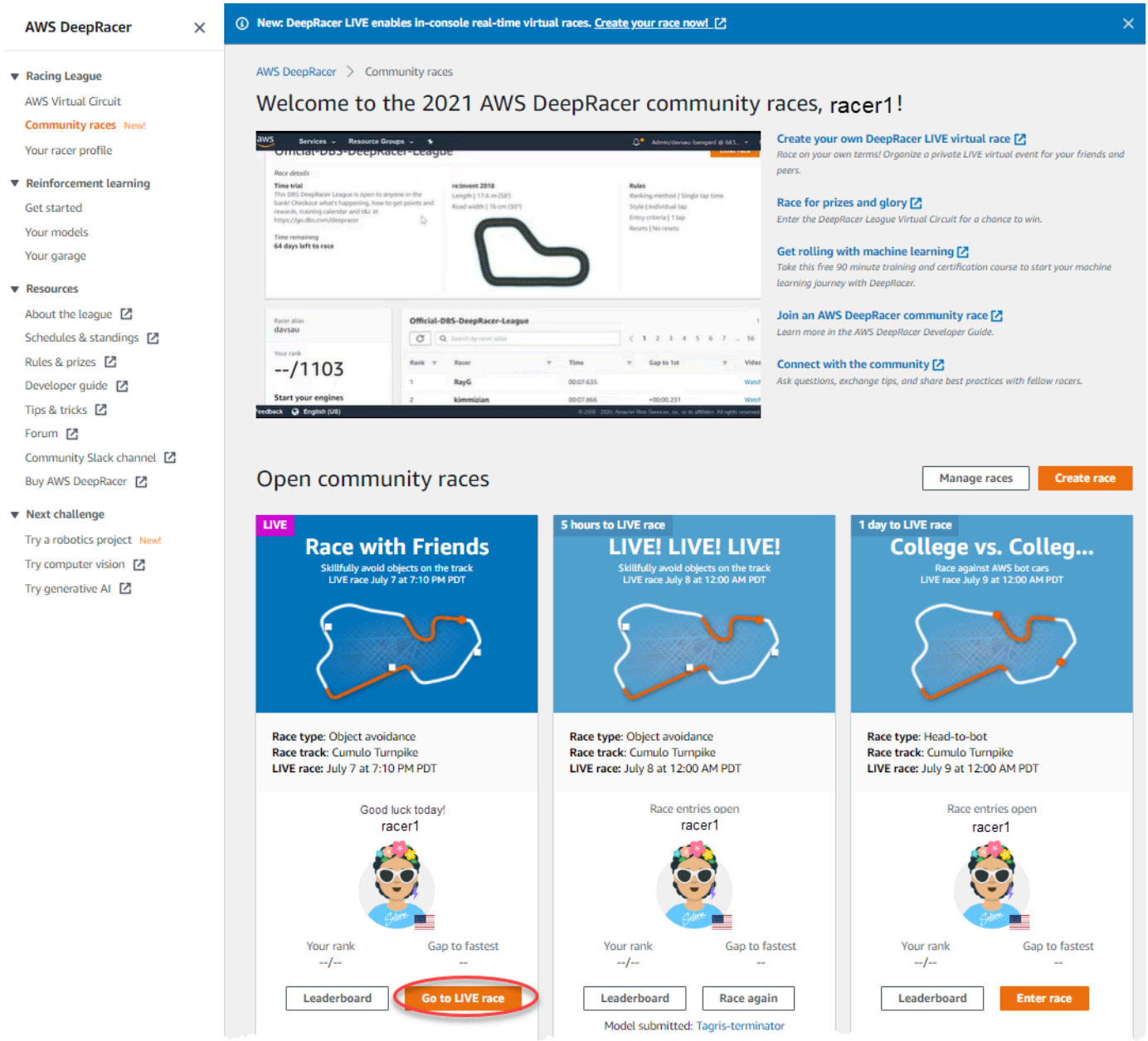
> **Note**
>
> 라이브 레이스 시작 시간 최소 1시간 전에 모델을 제출하십시오. 여러 모델이 출전할 수 있지만, 제출 창이 닫히기 전에 마지막으로 제출한 모델만 사용됩니다.

시작하기 전에

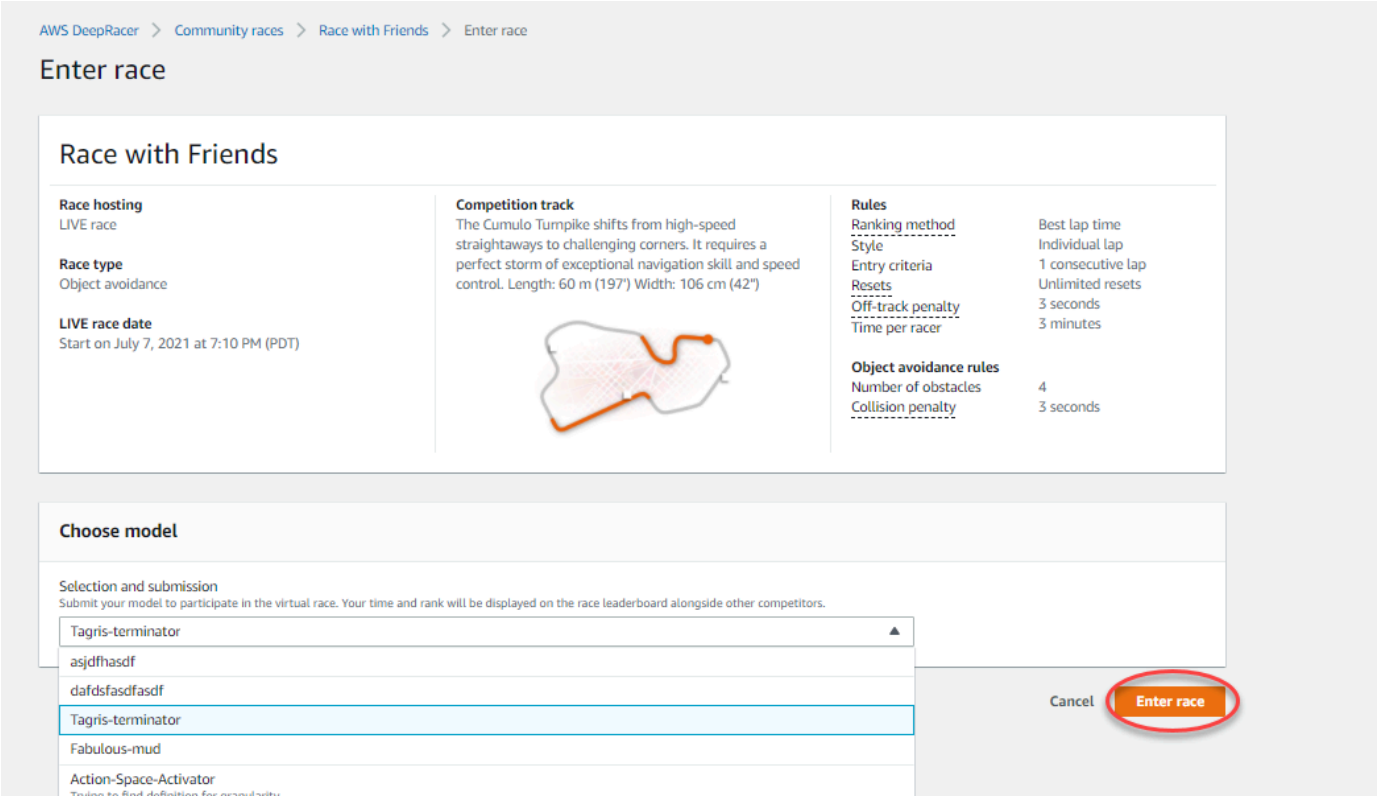
- Chrome 또는 Firefox 브라우저를 사용하십시오(브라우저가 최신인지 확인하십시오).
- 가상 프라이빗 네트워크(VPN)을 사용 중인 경우 연결을 끊으십시오.
- 추가 탭을 모두 닫습니다.

라이브 레이스에 참여하십시오.

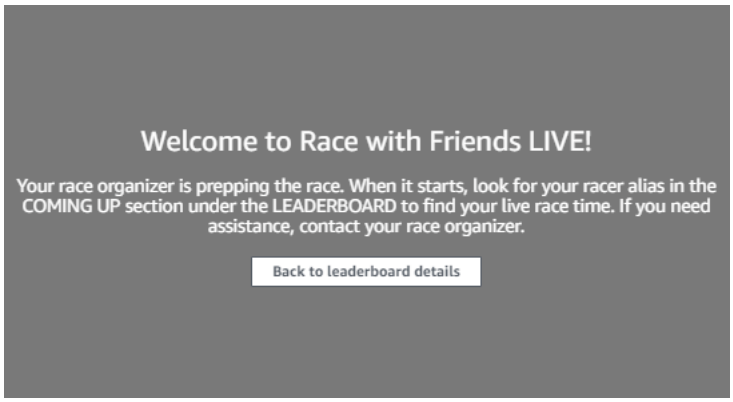
1. AWS DeepRacer 콘솔에 로그인합니다.
2. 모델을 제출하지 않은 경우 참가하려는 레이스의 레이스 카드를 찾아 라이브 레이스로 가기를 선택하십시오.



3. 레이스 페이지에서 레이스 참가를 선택합니다.
4. 레이스 참가 페이지의 모델 선택 아래 드롭다운 메뉴에서 출전하려는 모델을 선택하고 레이스 참가를 선택합니다.



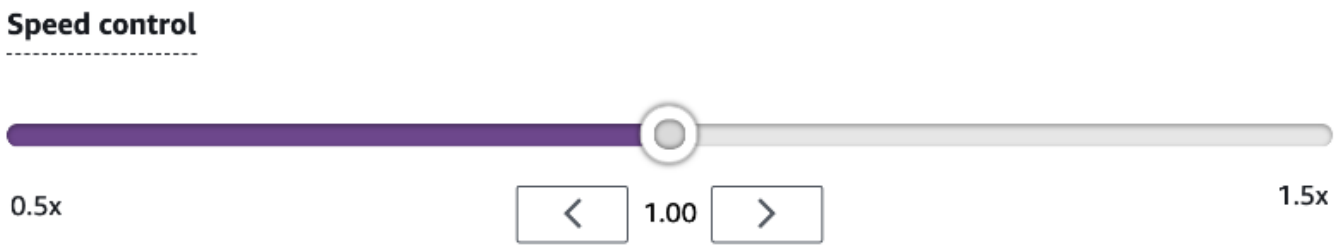
5. 레이스 페이지에서 라이브 레이스로 이동을 선택합니다.
6. 라이브 레이스 페이지에서 대기 메시지를 확인할 수 있습니다. 레이스 주최자가 제공한 컨퍼런스 브리지로 이동하십시오.



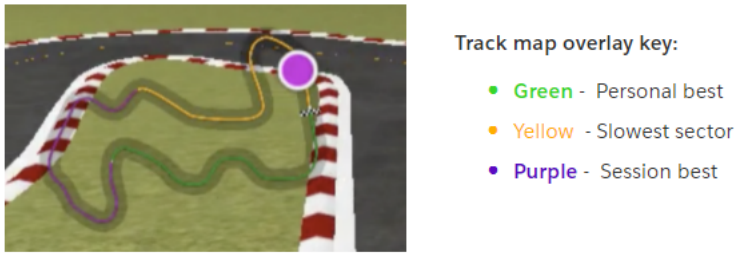
7. 레이스 주최자에게 문의하십시오. 주최자가 레이스 규칙을 검토하고 레이서의 질문에 답변해 드립니다.
8. 리더보드 아래 커밍 업 섹션에서 라이브 레이스 시간을 확인하고 레이스 주최자가 다음 순서라고 발표할 때 미리 준비하십시오.
9. 여러분의 차례가 되면 레이스 주최자가 레이스를 시작하면 콘솔에서 10, 9, 8, 7, 6... 카운트다운 애니메이션이 나타납니다. Go!에서 선택사양인 속도 제어를 이용할 수 있습니다. 모델 속도를 높

이거나 낮출 주요 순간을 선택할 수 있습니다. 속도 제어 기능을 작동하는 세 가지 방법이 있습니다.

a. 컴퓨터 마우스로 슬라이더를 드래그합니다.
b. 콘솔에서 < / > 화살표 버튼을 선택해도 됩니다.
c. 슬라이더 노브를 선택하여 슬라이더를 활성화한 다음 # 및 # 키보드 화살표 키를 사용할 수도 있습니다.



10. 모델의 속도 파라미터를 다시 사용하려면 멀티플라이어를 1로 재설정하십시오.
11. 레이스를 할 때는 라이브 레이스의 비디오 오버레이를 확인하여 퍼포먼스를 최적화하십시오. 트랙 맵 오버레이는 페이스에 따라 색이 변하는 세 개의 섹터로 나뉩니다. 녹색은 트랙에서 개인 최고 기록을 기록한 구간을, 노란색은 가장 느린 구간 구간을, 보라색은 세션 최고 기록을 나타냅니다. 또한 최고 랩 타임, 남은 시간, 속도(m/s), 리셋, 현재 랩 타임을 자세히 설명하는 통계를 확인할 수 있습니다.



12. 콘솔에 체크무늬 깃발 아이콘이 보이면 레이스가 끝납니다. 속도 제어가 비활성화되고 비디오 화면에서 레이스 리플레이가 시작됩니다. 최고 랩 타임 1개를 기준으로 리더보드 순위가 매겨집니다.

AWS DeepRacer 커뮤니티 레이스 주최

커뮤니티 레이스는 AWS가 공식적으로 후원하지 않는 AWS DeepRacer 사용자가 주최하는 레이스입니다.

자신만의 커뮤니티 레이스를 만들고 레이스 초대 링크를 공유하여 동료, 급우 또는 친구를 초대할 수 있습니다.

학생들을 위한 레이스를 조직하려면 AWS DeepRacer Student에 사용되는 교육자 도구를 참조하십시오.

주제

- the section called “레이스 빠른 시작 만들기”
- the section called “레이스를 사용자 지정하십시오.”
- the section called “라이브 레이스를 진행해 보십시오.”
- the section called “레이스 관리”
- the section called “레이싱 이벤트 용어”

가상 커뮤니티 레이스 만들기: 빠른 시작 안내서

기본 커뮤니티 레이스 설정을 사용하여 가상 레이스를 빠르게 설정할 수 있습니다. 모든 옵션에 대해 알아볼 준비가 되면 the section called “레이스를 사용자 지정하십시오.”로 이동합니다.

가상 레이스를 만들기 전에 클래식과 라이브 중 어떤 레이스가 그룹에 가장 적합한지, 그리고 라이브 레이스를 선택할 경우 비공개로 공유할지 아니면 공개적으로 공유할지 생각해 보십시오.

클래식 레이스

클래식 레이스는 실시간 상호 작용이 필요하지 않은 비동기 이벤트입니다. 참가자는 초대 링크를 받아야 레이스에 모델을 제출하고 리더보드를 확인할 수 있습니다. 레이서는 날짜 범위 내에서 언제든지 모델을 무제한으로 제출하여 리더보드에 오를 수 있습니다. 속도 제어는 불가능합니다. 제출된 모델의 클래식 레이스 결과 및 동영상은 레이스가 시작되자마자 리더보드 페이지에서 볼 수 있습니다. 모든 클래식 레이스는 비공개 이벤트입니다.

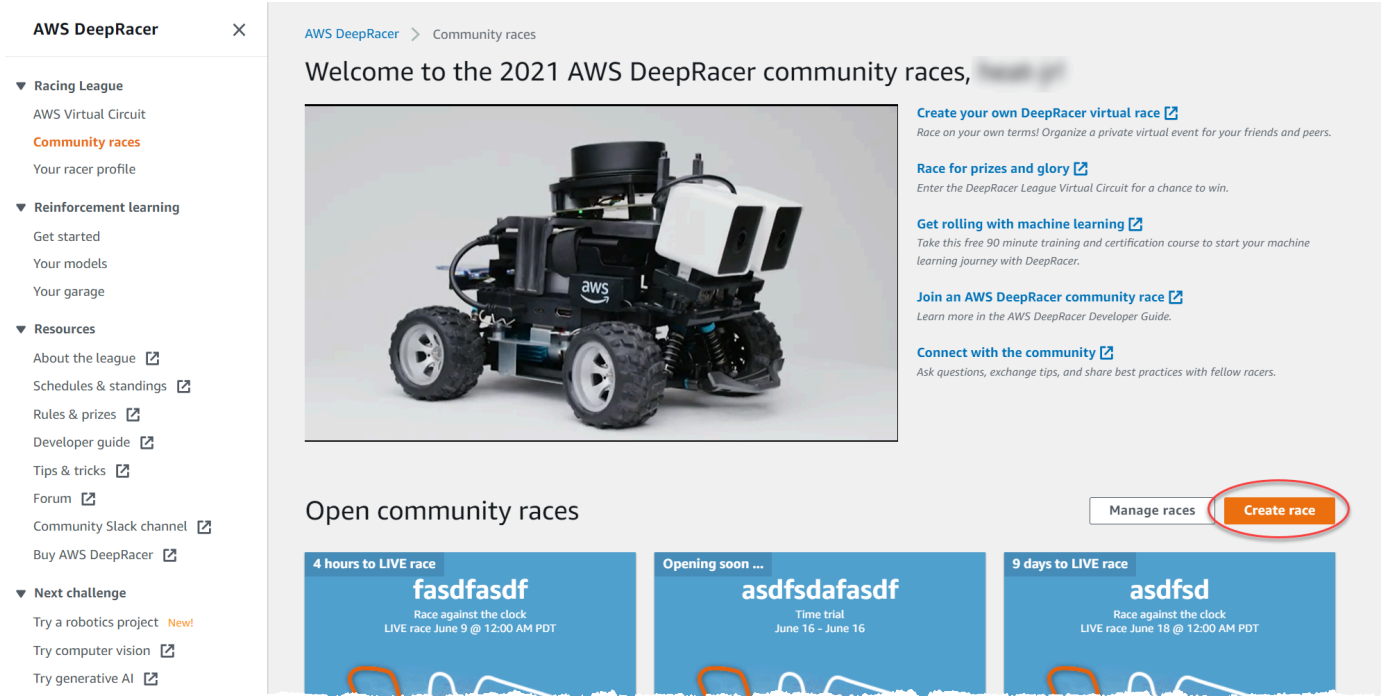
라이브 레이스

라이브 레이스는 정해진 시간에 개최되는 동기 이벤트로, 한 명의 레이스 주최자가 비공개 화상 회의를 진행하는 소규모 이벤트부터 주최자, 해설자, 방송사로 구성된 작은 팀이 공개적으로 중계하

는 대규모 이벤트까지 다양한 범위에서 진행됩니다. 언제든지 모델 제출 문을 열고 닫을 수 있으므로 레이서들에게 마감일을 알려주십시오. 참가자는 여러 모델을 제출할 수 있지만, 이벤트 기간 중에는 마감하기 전에 마지막으로 제출한 모델만 레이스에 참가할 수 있습니다. 라이브 레이스 중에 대기 중인 참가자는 대화형 속도 제어를 사용하여 자신의 차례에 모델이 경쟁 우위를 점할 수 있도록 할 수 있습니다. 라이브 레이스 참가자도 모델을 레이스에 제출할 수 있는 초대 링크를 받아야 하지만, 초대된 참가자에게만 이벤트를 비공개로 방송하거나 Twitch와 같은 라이브 스트리밍 서비스를 사용하여 공개적으로 이벤트를 중계할 수 있습니다. 자세한 내용은 the section called “레이스를 생중계하십시오.” 섹션을 참조하십시오.

커뮤니티 레이스 만들기 시작하기

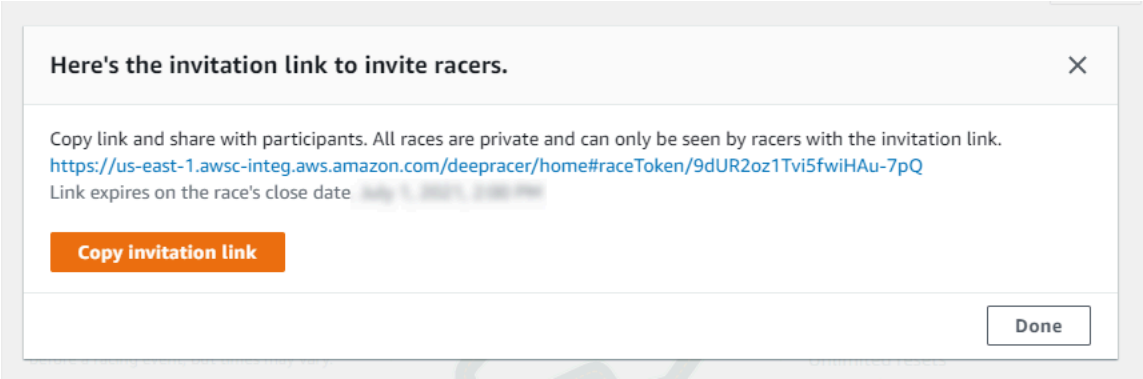
1. AWS DeepRacer 콘솔을 엽니다.
2. 커뮤니티 레이스를 선택합니다.
3. 커뮤니티 레이스 페이지에서 레이스 만들기를 선택합니다.



4. 레이스 세부 정보 페이지에서 경쟁 형식을 선택합니다. 즉, 고객이 설정한 시간 내에 자신의 일정에 따라 참가할 수 있는 클래식 레이스또는 실시간 이벤트로 비공개 또는 공개적으로 방송할 수 있는 라이브 레이스를 선택할 수 있습니다.

클래식 레이스를 계속 만들려면

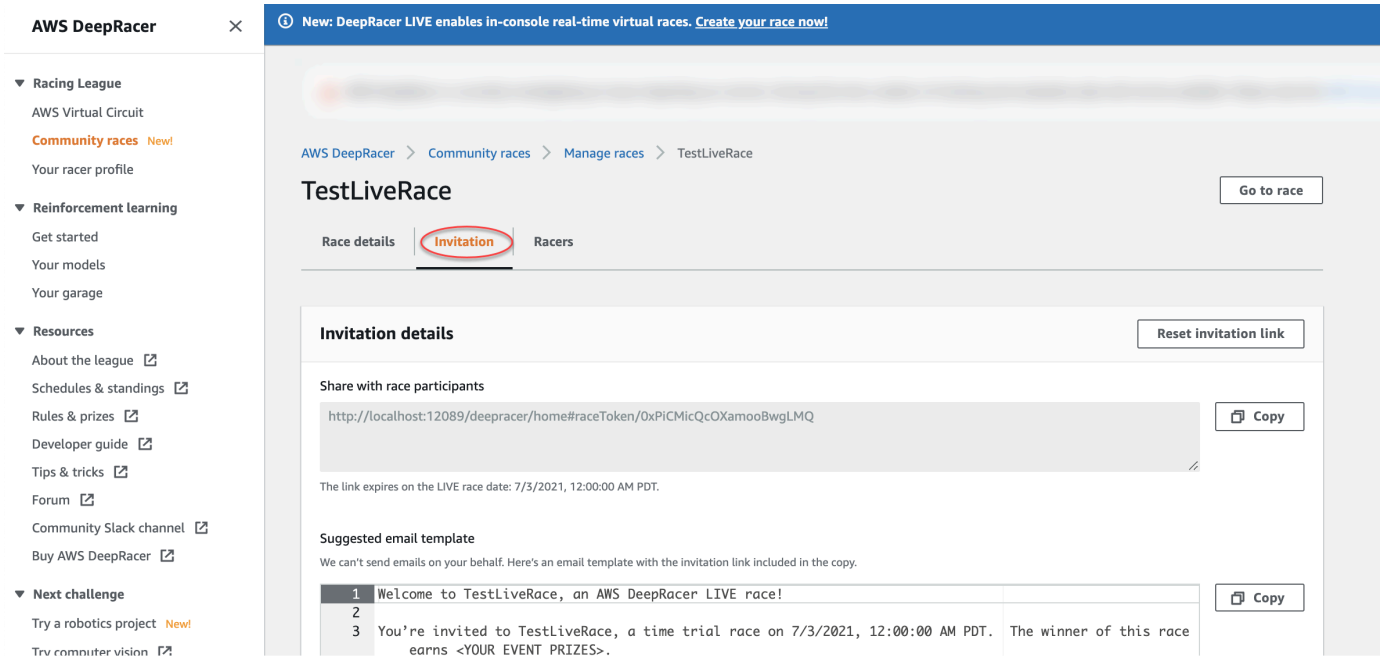
1. 레이스 유형을 선택합니다. 타임 트라이얼부터 장애물 회피, 헤드-투-봇의 순서로 레이스 유형이 점점 더 복잡해집니다. 레이스를 처음 하는 사람에게는 타임 트라이얼을 추천합니다. 타임 트라이얼 레이스는 카메라 한 대만 필요하기 때문에 센서 구성이 더 간단하고, 이런 유형의 레이스를 위해 강화형 기계 학습(RL) 모델이 더 빠르게 수렴됩니다. 레이스 유형에 대한 자세한 내용은 타임 트라이얼, 장애물 회피 및 헤드-투-봇 레이스를 위한 맞춤형 AWS DeepRacer 교육을 참조하십시오.
2. 레이스에 대한 설명이 포함된 원본 이름을 입력합니다.
3. 이벤트 시작 날짜 및 시간을 24시간 형식으로 지정합니다. AWS DeepRacer 콘솔은 사용자의 시간대를 자동으로 인식합니다. 클래식 레이스의 경우 종료 날짜 및 시간도 입력하십시오. 라이브 레이스의 기본 지속 시간은 4시간입니다. 더 긴 레이스를 예약하려면 고객 지원팀에 문의하십시오. 라이브 레이스가 일찍 종료되는 경우 취할 조치가 없습니다.
4. 다음을 선택하여 기본 설정을 사용합니다.
5. 레이스 세부 정보 검토 페이지에서 레이스 사양을 확인하십시오. 변경하려면 편집 또는 이전을 선택하여 레이스 세부 정보 페이지로 돌아가십시오. 초대 링크를 받을 준비가 되면 제출을 선택합니다.
6. 레이스를 공유하려면 모달에서 초대 링크 복사를 선택하고 이메일, 문자 메시지 및 즐겨 사용하는 소셜 미디어 애플리케이션에 붙여넣으십시오. 모든 클래식 레이스는 비공개로 진행되며 초대 링크를 받은 레이서만 관람할 수 있습니다. 링크는 레이스 마감일에 만료됩니다.



7. 완료 를 선택합니다. 레이스 관리 페이지가 표시됩니다.
8. 클래식 레이스 기간이 끝나면 리더보드 세부 정보 페이지의 레이서 아래에 누가 모델을 제출했는지, 누가 아직 모델을 제출해야 하는지 기록해 두십시오.

라이브 레이스를 계속 만들려면

1. 레이스 유형을 선택합니다. 타임 트라이얼부터 장애물 회피, 헤드-투-봇의 순서로 레이스 유형이 점점 더 복잡해집니다. 레이스를 처음 하는 사람에게는 타임 트라이얼을 추천합니다. 타임 트라이얼 레이스는 카메라 한 대만 필요하기 때문에 센서 구성이 더 간단하고, 이런 유형의 레이스를 위해 강화형 기계 학습(RL) 모델이 더 빠르게 수렴됩니다. 레이스 유형에 대한 자세한 내용은 타임 트라이얼, 장애물 회피 및 헤드-투-봇 레이스를 위한 맞춤형 AWS DeepRacer 교육을 참조하십시오.
2. 레이스에 대한 설명이 포함된 원본 이름을 입력합니다.
3. 이벤트 시작 날짜 및 시간을 24시간 형식으로 지정합니다. AWS DeepRacer 콘솔은 사용자의 시간대를 자동으로 인식합니다. 클래식 레이스의 경우 종료 날짜 및 시간도 입력하십시오. 라이브 레이스의 기본 지속 시간은 4시간입니다. 더 긴 레이스를 예약하려면 고객 지원팀에 문의하십시오. 라이브 레이스가 일찍 종료되는 경우 취할 조치가 없습니다.
4. 다음을 선택하여 기본 설정을 사용합니다.
5. 레이스 세부 정보 검토 페이지에서 레이스 사양을 확인하십시오. 변경하려면 편집 또는 이전을 선택하여 레이스 세부 정보 페이지로 돌아가십시오. 초대 링크를 받을 준비가 되면 제출을 선택합니다.
6. <Your Race Name> 페이지에서 초대 탭을 선택하여 레이스를 공유하십시오.



7. 초대 세부 정보에서 복사를 선택하여 초대 링크를 이메일, 문자 메시지 및 즐겨 사용하는 소셜 미디어 애플리케이션에 붙여넣습니다.

8. 선택 사항으로 제안된 이메일 템플릿 옆의 복사를 선택하고 상품, 모델 제출 기간, 레이서들이 대기열에 등록하고 레이스를 준비하는 모임 장소인 컨퍼런스 브리지 링크를 입력합니다.

   라이브 레이스는 비공개로 진행되며, 공개적으로 방송하도록 선택하지 않는 한 초대 링크를 가진 레이서들만 시청할 수 있습니다. 자세한 내용은 the section called “레이스를 생중계하십시오.” 섹션을 참조하십시오. 링크는 레이스 마감일 오전 12시(PDT)에 만료됩니다.

9. 레이스 세부 정보 탭을 선택합니다.

10. 레이스 세부 정보에서 라이브 레이스 중계 옵션을 참고하십시오. 레이스를 공개적으로 방송할지 비공개로 방송할지 결정했으면 AWS DeepRacer League 팀에서 만든 플레이북을 사용하여 시작하십시오. 브로드캐스트 모드 보기 버튼을 사용하면 해설자 스트리밍을 위한 컷아웃이 포함된 브랜드 그래픽 오버레이와 함께 사용할 수 있도록 형식이 지정된 라이브 레이스 이벤트 페이지를 볼 수 있습니다.

11. 라이브 레이스 날짜가 다가오면 <Your Race Name> 페이지의 초대 탭 아래에 누가 모델을 제출했는지, 누가 아직 참가해야 하는지 기록해 두십시오.

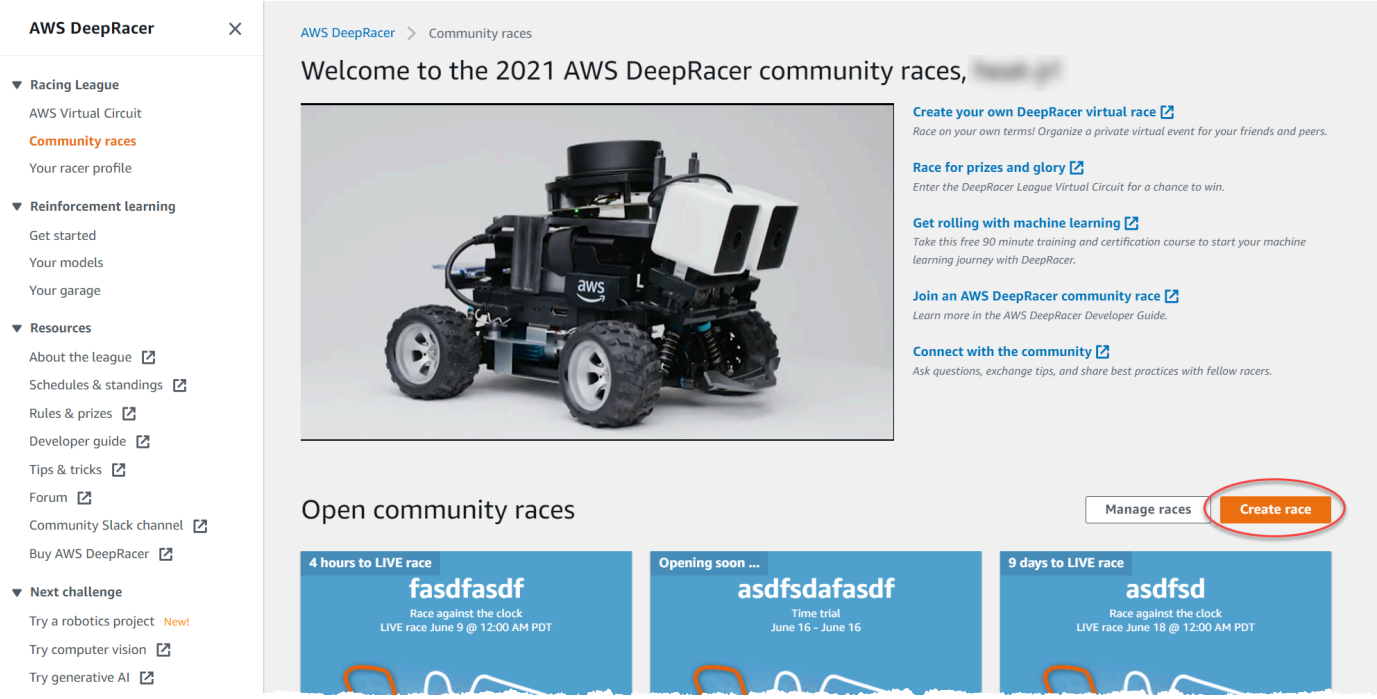
선택한 레이스 트랙을 변경하고, 레이스 설명을 추가하고, 순위 지정 방법을 선택하고, 레이서를 재설정할 수 있는 횟수를 결정하고, RL 모델이 레이스에 출전하기 위해 완료해야 하는 최소 랩 수를 결정하고, 오프 트랙 페널티를 설정하고, 기타 레이스 세부 정보를 사용자 지정하려면 커뮤니티 레이스 관리에서 레이스 세부 정보 편집을 선택합니다.

레이스를 사용자 지정하십시오.

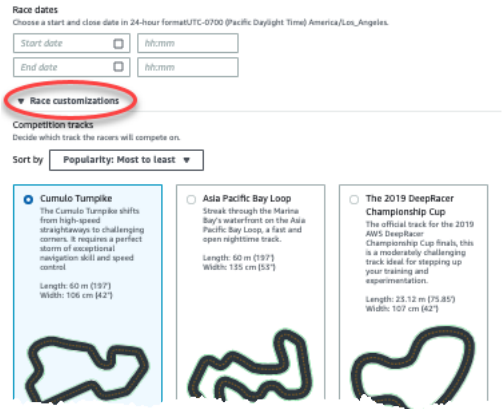
그룹에 맞는 레이스를 만들려면 레이스 세부 정보 페이지에서 레이스 사용자 지정을 확장하십시오. 타임 트라이얼 레이스 설정은 장애물 회피 및 헤드-투-봇 레이스에도 적용되지만, 장애물 회피 및 헤드-투-봇 레이스 유형에는 이벤트 목표에 맞게 특별히 조정된 레이스 환경을 만들 수 있는 추가 설정이 있습니다.

레이스를 사용자 지정하려면

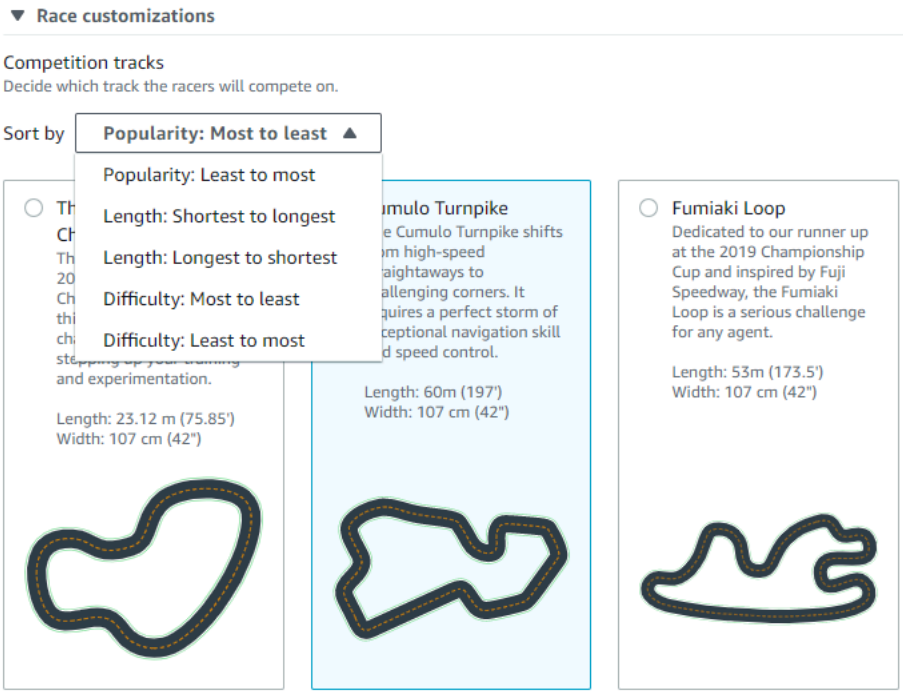
1. AWS DeepRacer 콘솔을 엽니다.
2. 커뮤니티 레이스를 선택합니다.
3. 커뮤니티 레이스 페이지에서 레이스 만들기를 선택합니다.



4. 레이스 세부 정보 페이지에서 경쟁 형식을 선택합니다. 즉, 설정한 시간 내에 자신의 일정에 따라 참가할 수 있는 클래식 레이스또는 실시간 이벤트로 비공개 또는 공개로 방송할 수 있는 라이브 레이스를 선택할 수 있습니다.
5. 선택한 대회 형식에 따라 클래식 레이스를 계속생성하려면 또는 the section called “레이스 빠른 시작 만들기”에서 라이브 레이스를 계속 만들려면의 1~3단계를 따르십시오.
6. 레이스 날짜를 선택한 후 레이스 사용자 지정을 확장하십시오.

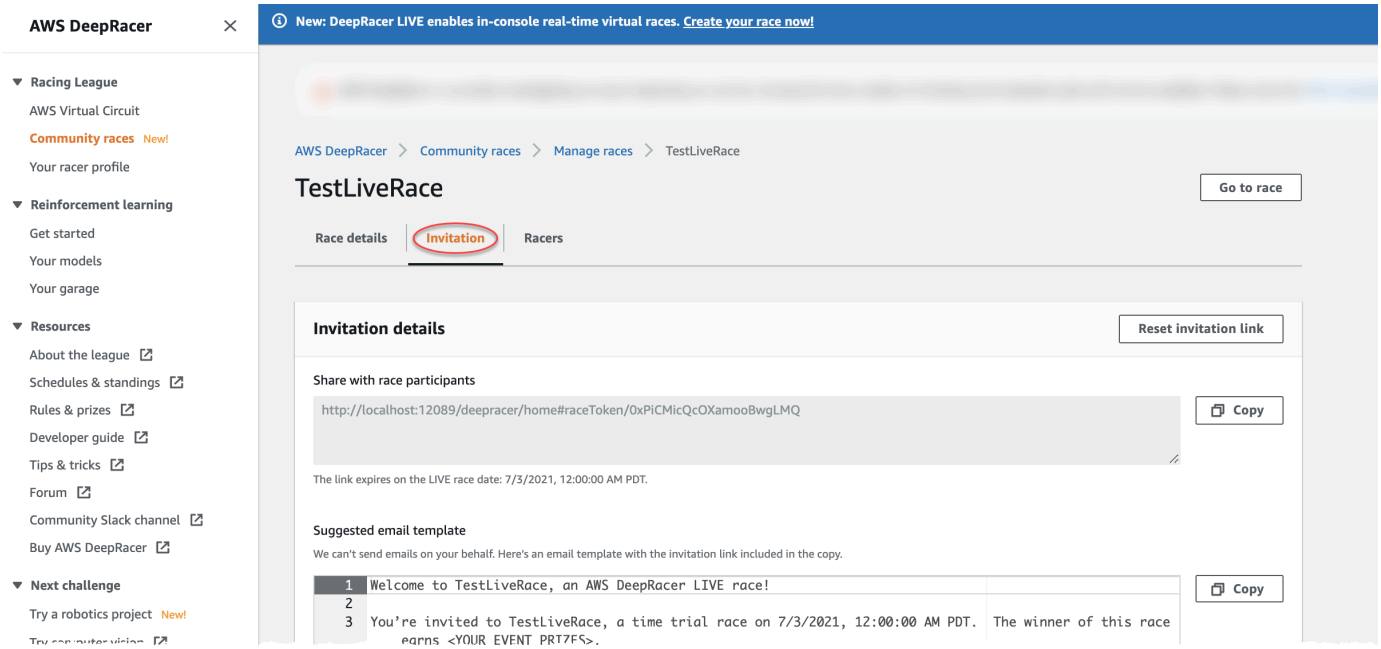


7. 경쟁 트랙을 선택합니다. 인기도: 가장 높은 것부터 낮은 것/가장 낮은 것부터 가장 높은 것, 난이도: 가장 높은 것부터 낮은 것/가장 낮은 것부터 가장 높은 것, 길이: 가장 긴 것부터 가장 짧은 것/가장 짧은 것부터 가장 긴 것 순으로 트랙을 정렬할 수 있습니다. 각 카테고리의 모든 트랙을 보려면 더 많은 레이스 트랙 옵션 보기를 선택하십시오. 확장된 메뉴를 닫으려면 더 적은 레이스 트랙 옵션 보기를 선택합니다.



8. 참가자를 위해 이벤트의 목표와 규칙을 요약한 레이스에 대한 설명을 작성할 수도 있습니다. 라이브 레이스의 경우 이벤트의 화상 회의 또는 라이브 스트림을 위한 링크를 추가하십시오. 설명은 리더보드 세부 정보에 표시됩니다.
9. 클래식 레이스의 랭킹 방식은 가장 빠른 랩을 기록한 레이서가 우승하는 베스트 랩 타임, 이벤트 기간 내에 여러 번 시도한 결과 평균시간이 가장 좋은 레이서가 우승하는 평균 시간, 전체 평균 시간이 가장 빠른 레이서가 우승하는 총 시간 중에서 선택하십시오. 모든 라이브 레이스의 리더보드 순위는 최고 랩 시간을 기준으로 순위가 매겨지므로 이 필드는 표시되지 않습니다.
10. 클래식 레이스의 경우 최소 랩 값을 선택하십시오. 최소 랩 수는 레이서가 레이스의 리더보드에 결과를 제출할 자격을 갖추기 위해 완료해야 하는 연속 랩 수입니다. 초보자 레이스의 경우 작은 숫자를 선택합니다. 고급 사용자의 경우 더 큰 숫자를 선택합니다. 기본값은 랩 한 바퀴이므로 라이브 레이스에서는 이 사용자 지정을 사용할 수 없습니다.
11. 오프 트랙 페널티의 경우 RL 모델이 트랙을 벗어날 때 레이서의 시간에 추가할 시간을 초 단위로 선택합니다.
12. 이제 타임 트라이얼 레이스의 모든 사용자 지정 옵션을 완료했습니다. 타임 트라이얼 레이스 형식을 선택한 경우 다음을 선택하여 레이스 세부 정보를 검토하십시오. 장애물 회피 또는 헤드-투-봇 레이스 형식을 선택한 경우 적절한 절차로 건너뛰어 레이스 사용자 지정을 완료하십시오.
13. 레이스 세부 정보 검토 페이지에서 레이스 사양을 검토합니다. 변경하려면 편집 또는 이전을 선택하여 레이스 세부 정보 페이지로 돌아가십시오. 초대 링크를 받을 준비가 되면 제출을 선택합니다.

14. 레이스를 공유하려면 모달의 초대 링크를 클립보드에 복사하기를 선택하고 이메일, 문자 메시지, 즐겨 사용하는 소셜 미디어 애플리케이션에 붙여넣으십시오. 초대 탭을 선택하여 <Your Race Name> 페이지에서 레이스를 공유할 수도 있습니다. 링크는 레이스 마감일에 만료됩니다.

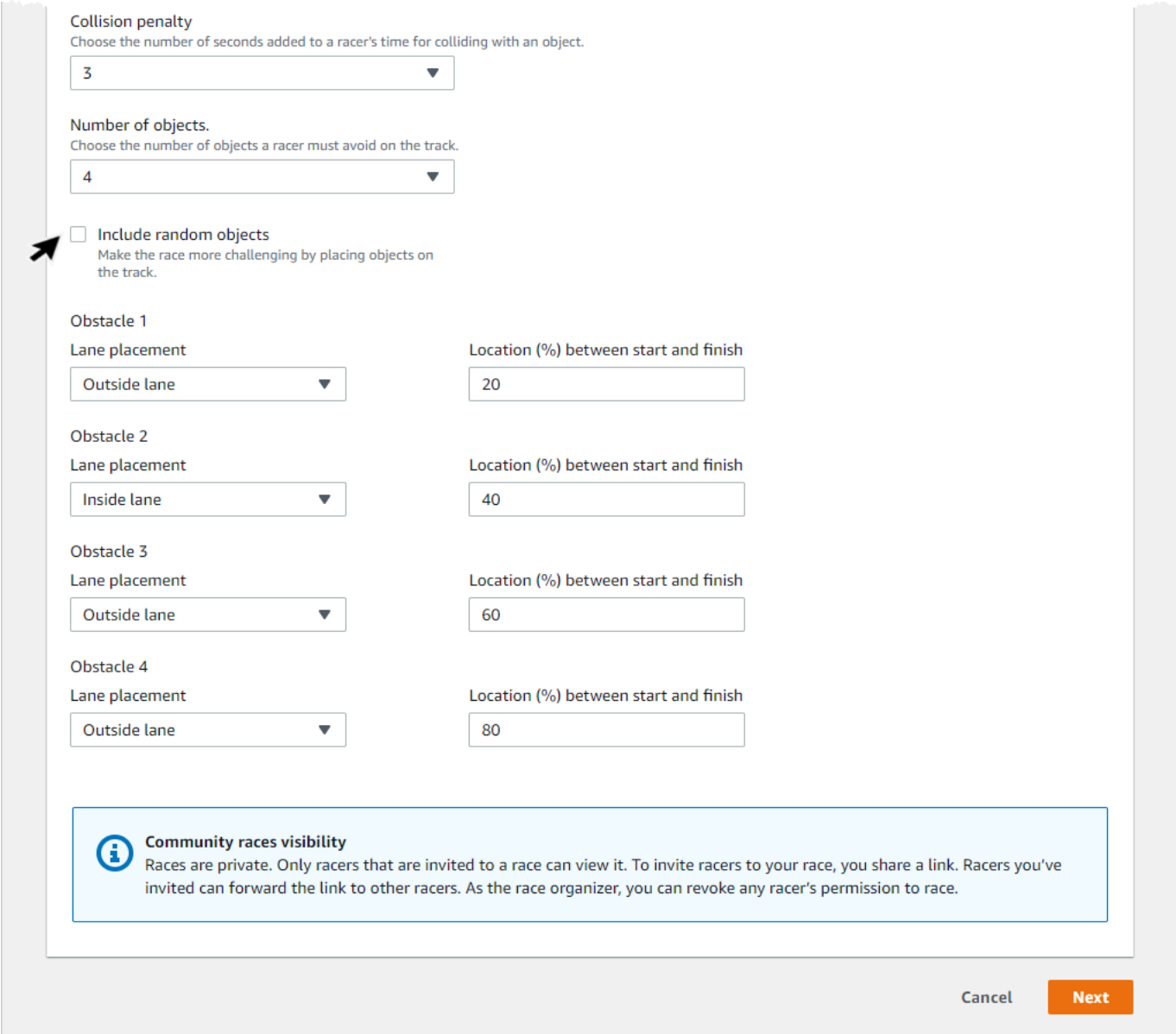


15. 완료 를 선택합니다. 레이스 관리 페이지가 표시됩니다.

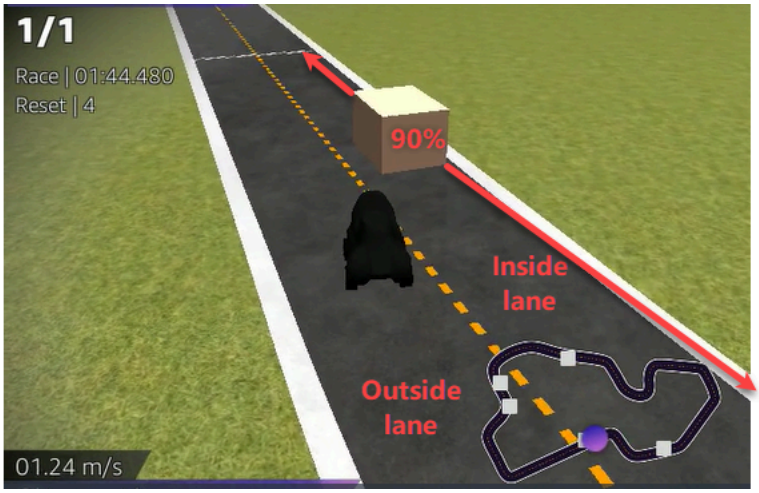
이메일 템플릿을 사용하여 새 레이서를 초대하고, 레이스에서 레이서를 삭제하고, 레이스의 모델 제출 상태 등을 확인하는 방법을 알아보려면 커뮤니티 레이스 관리를 참조합니다.

장애물 회피 레이스 사용자 지정을 마치려면

1. 충돌 페널티에서 레이서가 물체 또는 봇과 충돌하는 데 걸리는 시간에 추가되는 시간(초)을 선택하십시오. 몇 초가 추가될수록 도전 과제는 더 커집니다.



2. 장애물 수에서는 레이서가 트랙에서 피해야 하는 장애물의 갯수를 선택합니다. 장애물이 많을수록 레이스가 더 어려워집니다.
3. 레이서마다 다른 위치에 배치될 무작위 장애물을 레이스 트랙에 추가하려면 무작위 장애물 포함을 선택합니다. 레이스 트랙에서 예상치 못한 장애물과 같은 무작위 이벤트에 잘 일반화하는 RL 모델을 만들려면 더 오랜 시간 동안 훈련하고 보상 함수 시행착오를 겪어야 하기 때문에 참가자에게는 더 어려운 과제입니다.
4. 차선 배치에 사용할 차선 번호 또는 장애물 위치를 선택하여 각 장애물을 배치할 위치를 선택하십시오. 트랙은 중앙선에서 반으로 나뉘어 안쪽 및 바깥쪽 차선을 만듭니다. 안쪽 차선 또는 바깥쪽 차선에 장애물을 배치할 수 있습니다.

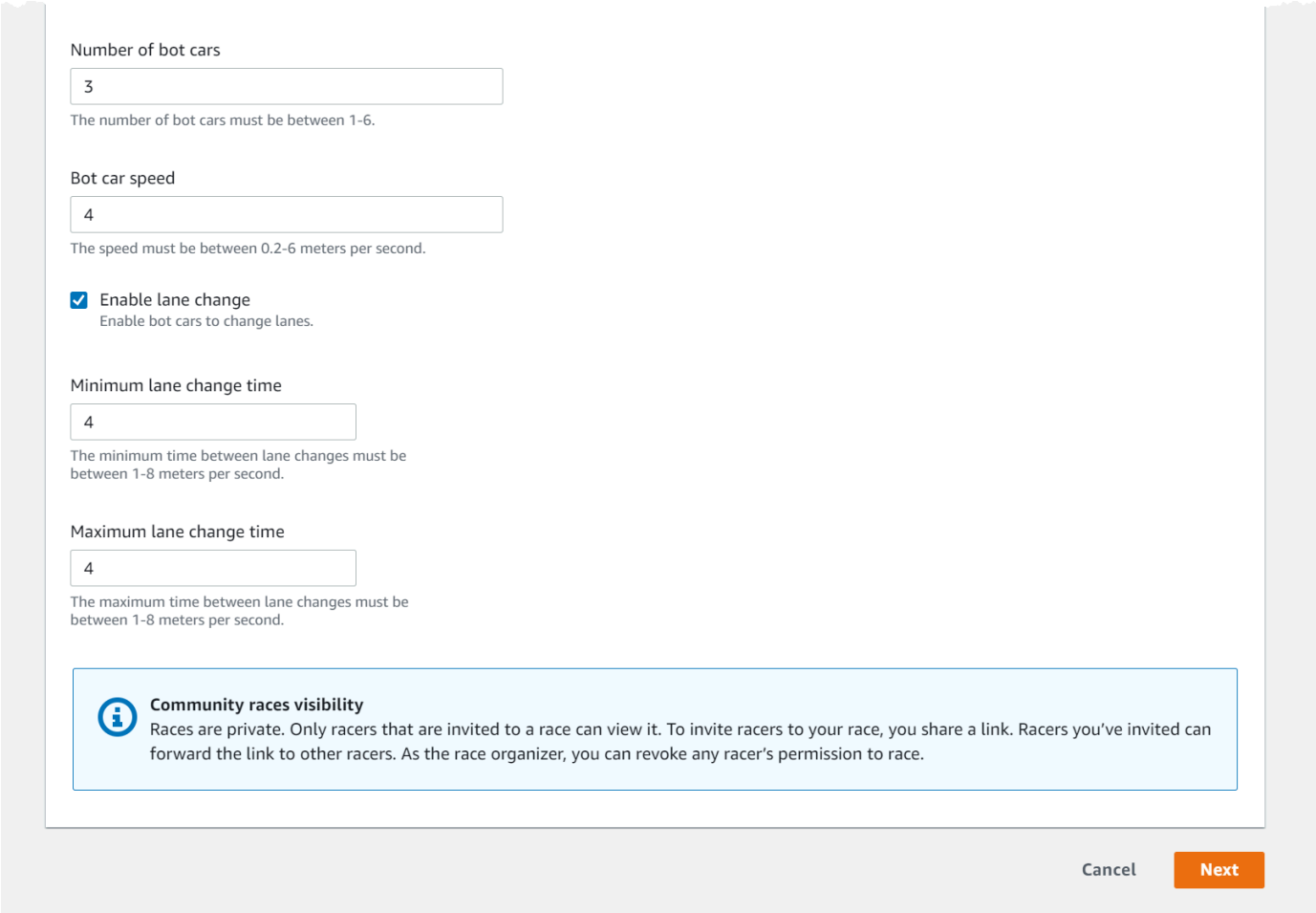


5. 각 장애물에 대해 시작과 종료 사이의 위치(%) 값을 선택합니다. 숫자는 장애물을 놓으려는 트랙의 시작선과 끝선 사이의 위치(백분율로 표시됨) 를 나타냅니다.
6. 이제 장애물 회피 레이스를 위한 고유한 사용자 지정 옵션을 모두 완료했습니다. 다음을 선택합니다.
7. 레이스 세부 정보 검토 페이지에서 레이스 사양을 검토합니다. 변경하려면 편집 또는 이전을 선택하여 레이스 세부 정보 페이지로 돌아가십시오. 초대 링크를 받을 준비가 되면 제출을 선택합니다.
8. 레이스를 공유하려면 초대 링크 복사를 선택하고 이메일, 문자 메시지 및 즐겨 사용하는 소셜 미디어 애플리케이션에 붙여넣으십시오. 모든 레이스는 비공개로 진행되며 초대 링크를 받은 레이서만 관람할 수 있습니다. 링크는 레이스 마감일에 만료됩니다.
9. 완료 를 선택합니다. 레이스 관리 페이지가 표시됩니다.

레이스로 무엇을 할 수 있는지 알아보려면 커뮤니티 레이스 관리를 참조하십시오.

헤드-투-봇 레이스 사용자 지정을 마치려면

1. 봇 자동차 수에서 참가자의 AWS DeepRacer RL 모델을 상대로 경주할 자동차 수를 선택합니다. 봇 자동차는 비디오 게임 AI 차량과 비슷합니다. 무작위로 움직이는 물체이기 때문에 고정된 물체보다 복잡성이 한 단계 더 높습니다. 트랙에 봇이 많을수록 레이스가 어려워집니다. 최대 6개까지 선택할 수 있습니다.



2. 봇 자동차 속도의 경우 봇 자동차가 트랙에서 얼마나 빨리 움직일지 선택하십시오. 속도는 초당 미터 단위로 측정됩니다. 속도는 초당 0.2~6미터여야 합니다.
3. 봇이 차선을 변경하도록 허용하여 레이서의 AWS DeepRacer RL 모델의 챌린지가 더욱 복잡해지도록 하려면 차선 변경 활성화를 선택하십시오.
4. 최소 차선 변경 시간에서 봇 차량이 차선을 변경하는 인스턴스 사이에 걸리는 최대 시간(초)을 선택합니다.
5. 최대 차선 변경 시간에서 봇 차량이 차선을 변경하는 인스턴스 사이에 걸리는 최대 시간(초)을 선택합니다.
6. 이제 헤드-투-봇 레이스를 위한 고유한 사용자 지정 옵션을 모두 완료했습니다. Next(다음)를 선택합니다.
7. 레이스 세부 정보 검토 페이지에서 레이스 사양을 검토합니다. 변경하려면 편집 또는 이전을 선택하여 레이스 세부 정보 페이지로 돌아가십시오. 초대 링크를 받을 준비가 되면 제출을 선택합니다.
8. 레이스를 공유하려면 초대 링크 복사를 선택하고 이메일, 문자 메시지 및 즐겨 사용하는 소셜 미디어 애플리케이션에 붙여넣으십시오. 모든 레이스는 비공개로 진행되며 초대 링크를 받은 레이서만 관람할 수 있습니다. 링크는 레이스 마감일에 만료됩니다.

9. 완료 를 선택합니다. 레이스 관리 페이지가 표시됩니다.

레이스를 편집하고 지우는 방법에 대해 알아보려면 커뮤니티 레이스 관리를 참조하십시오.

라이브 AWS DeepRacer 커뮤니티 레이스 진행

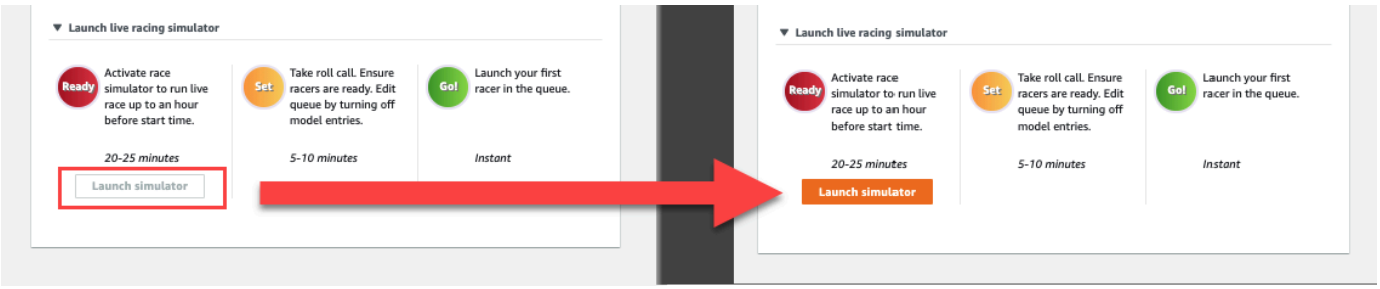
라이브 레이스를 만들고 레이서를 초대했습니다. 이벤트를 비공개로 방송할지 아니면 the section called "레이스를 생중계하십시오."의 지원을 받아 공개적으로 방송할지 결정했습니다. 이제 대기열을 관리하고, 레이스 시뮬레이터를 설정하고, 레이서를 실행하는 방법을 알아보십시오.

시작하기 전에

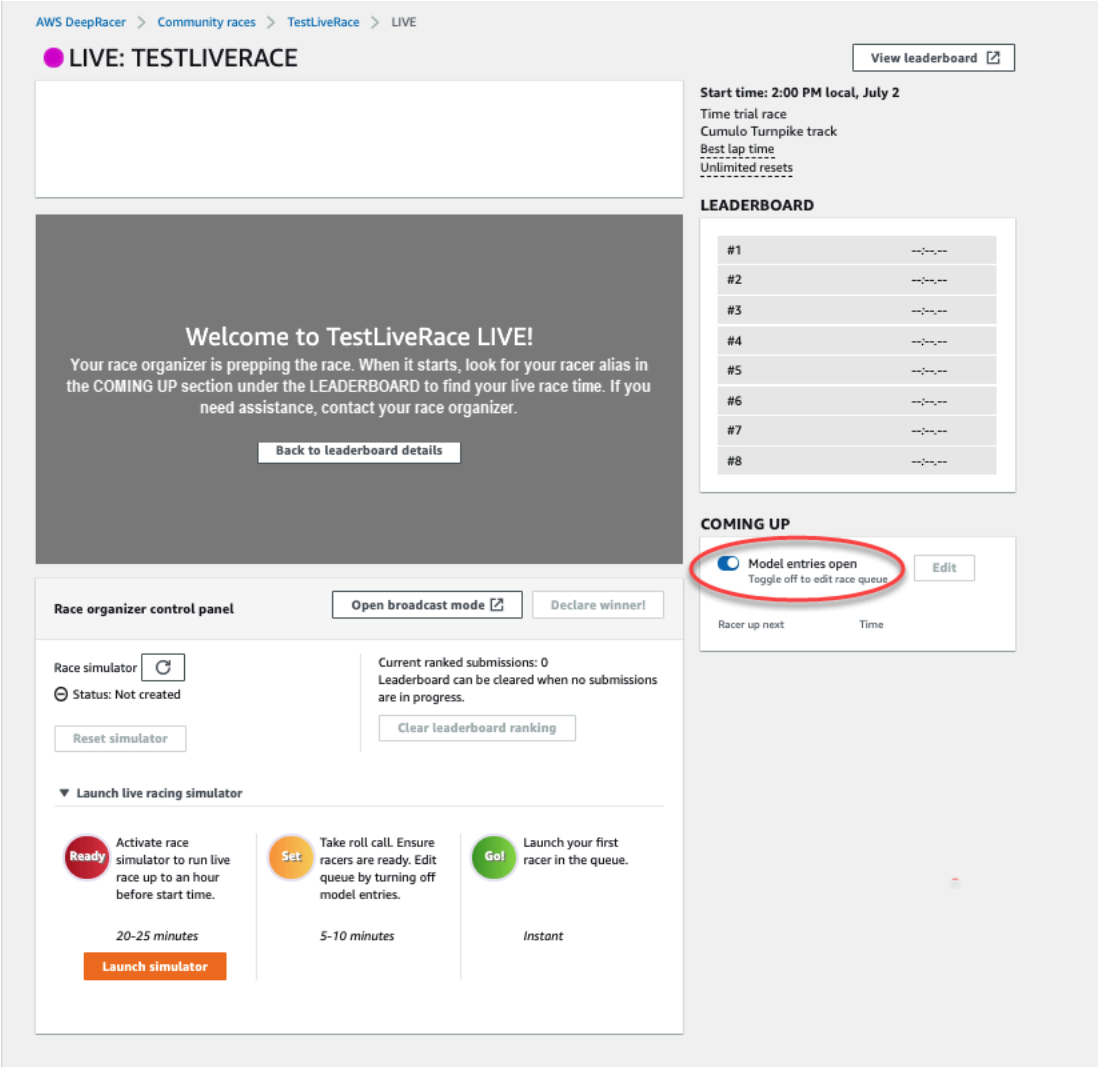
- Chrome 또는 Firefox 브라우저를 사용하십시오(브라우저가 최신인지 확인하십시오).
- 가상 프라이빗 네트워크(VPN)을 사용 중인 경우 연결을 끊으십시오.
- 추가 탭을 모두 닫습니다.

라이브 가상 레이스를 진행하려면

1. 커뮤니티 레이스 페이지에서 조정하려는 레이스의 레이스 카드를 찾은 다음 지금 가입을 선택하여 레이스를 확인하십시오.
2. 라이브: <Your Race Name> 페이지의 레이스 주최자 컨트롤 패널에서 시뮬레이터 실행을 선택합니다. 이 버튼은 레이스 시작 시간 한 시간 전에 사용할 수 있습니다. 라이브 레이싱 시뮬레이터 실행 제목을 선택하여 레이스 주최자 컨트롤 패널의 이 섹션을 숨길 수 있습니다.

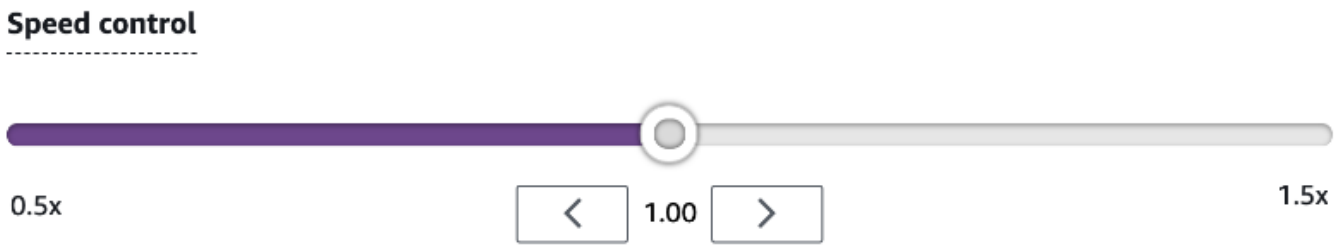


3. COMING UP에서 제출을 마감하기 위해 열려 있는 모델 항목을 끄십시오. 그러면 모델 제출이 마감되고 토글 아래에 편집 가능한 레이서 대기열이 생성됩니다. 토글을 끄기 전까지는 레이서를 출발시킬 수 없습니다.

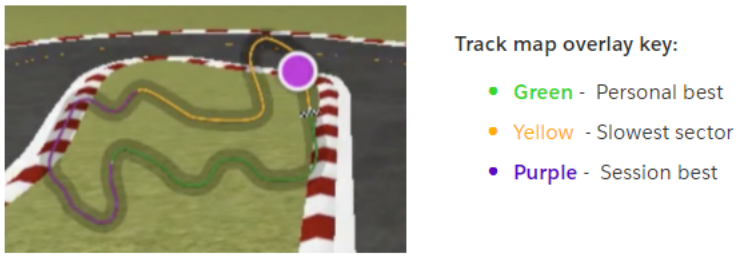


4. 만든 화상 회의를 열어 레이서들을 모으십시오.
5. 레이서 출석 호출 시작:
   a. 레이서들에게 문의하여 상대방이 귀하의 목소리를 잘 들을 수 있는지 확인하십시오.
   b. 처음에는 비디오를 사용하여 자신을 소개하십시오. 대역폭을 최적화하기 위해 나중에 종료할 수도 있습니다.
   c. 호출 중인 사람 목록이 그룹 내 레이서 목록과 일치하는지 확인하십시오.
6. 모델 출석 호출 시작:
   a. 레이서 대기열의 별칭 목록이 레이서의 별칭과 일치하는지, 빨간색으로 강조 표시된 별칭(모델이 제대로 제출되지 않았음을 의미)이 없는지 확인하십시오.
   b. 레이서들에게 문의하여 모델을 제출하는 데 문제가 있는지 확인하십시오.

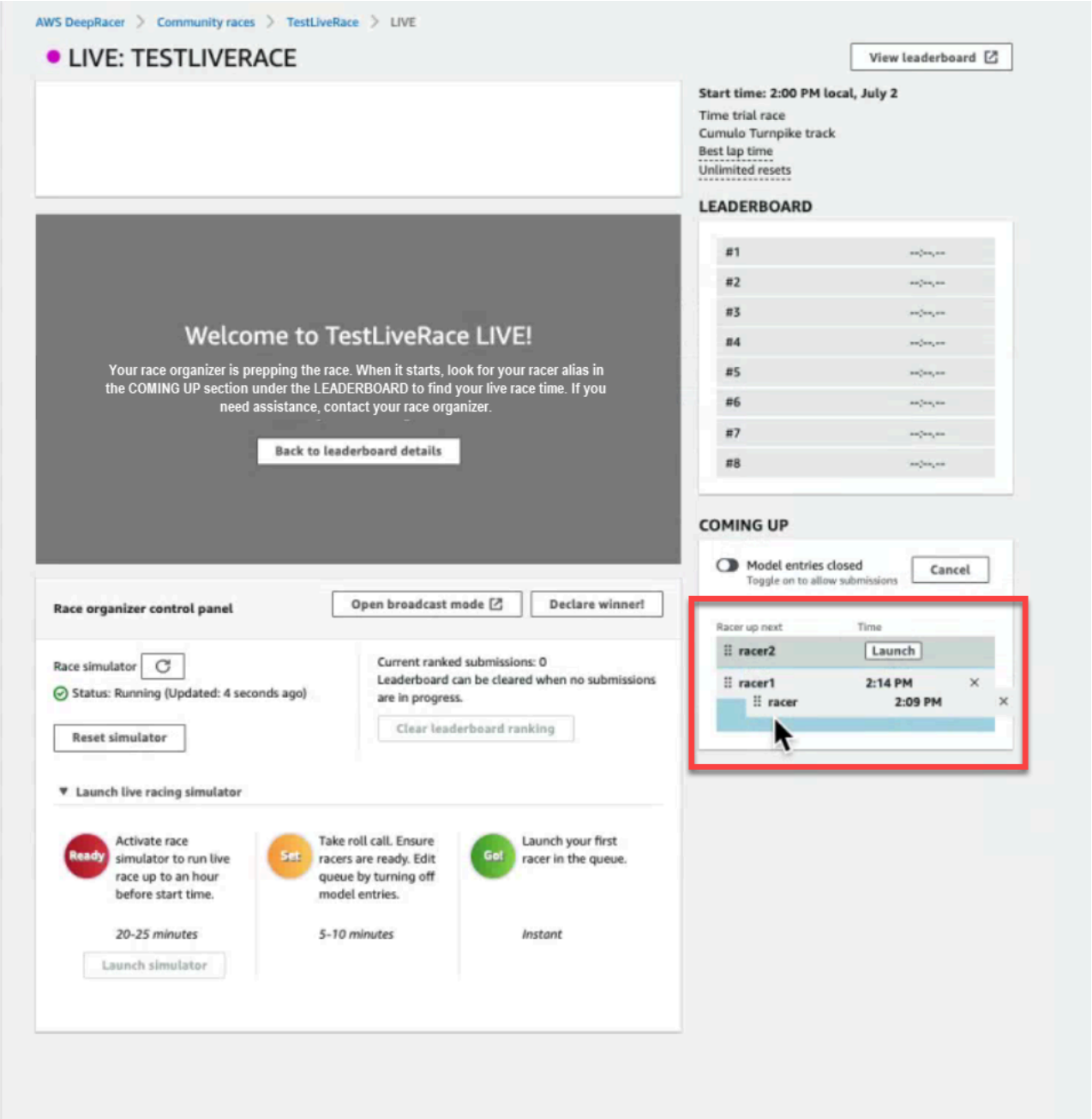
7. 레이스 일정과 규칙을 검토하십시오. 레이서들에게 자신의 차례에 얼마나 많은 시간을 할애해야 하는지 알려주고, 리더보드 순위는 해당 기간 동안 가장 빠른 단일 랩으로 결정된다는 점을 상기시키십시오.
8. 레이스 중에만 레이서에게 보이는 속도 제어 기능을 사용하면 속도 제어 슬라이더를 사용하여 수동으로 최대 속도를 설정할 수 있다고 설명하십시오. 속도 제어 슬라이더는 모델의 속도 파라미터를 일시적으로 무시하지만 조향 각도는 재정의하지 않습니다. 모델은 여전히 조향하지만 레이서들은 이제 속도를 곱하여 자동차 속도를 높이거나 낮출 주요 순간을 선택할 수 있습니다. 레이서들은 모델의 속도 파라미터를 다시 사용하기 위해 멀티플라이어를 1로 재설정할 수 있습니다. 스피드 컨트롤 슬라이더는 가속 페달이 아니라 전략적인 실시간 조정을 위한 기회라는 점을 레이서들에게 상기시켜 주십시오.



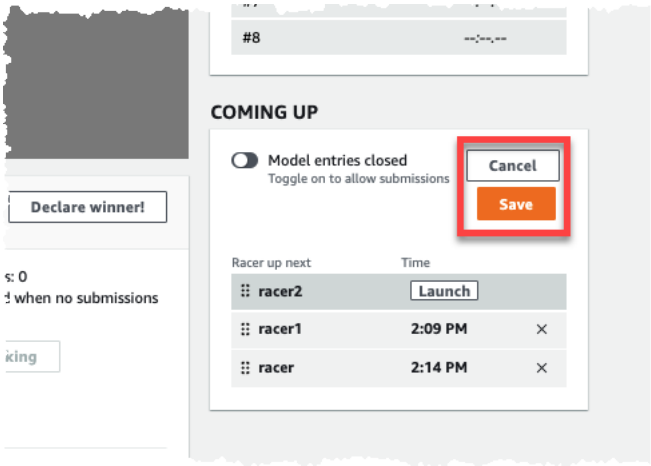
9. 그 다음, 레이스 창의 비디오 오버레이에는 레이서의 성능을 최적화하는 데 도움이 되는 정보가 포함되어 있다고 설명합니다. 트랙 맵 오버레이는 페이스에 따라 색이 변하는 세 개의 섹터로 나뉩니다. 녹색은 트랙에서 개인 최고 기록을 기록한 구간을, 노란색은 가장 느린 구간 구간을, 보라색은 세션 최고 기록을 나타냅니다. 또한 레이서들은 자신의 최고 랩 타임, 남은 속도(m/s), 리셋, 현재 랩 타임을 자세히 설명하는 통계를 확인할 수 있습니다.



10. 레이서 질문에 답하십시오.
11. 선택 사항으로 COMING UP에서 편집을 선택하여 레이서 이름을 가져다 놓아 레이스 대기열을 재정렬할 수 있습니다.



12. 레이서 대기열을 변경한 경우 저장을 선택하여 편집 내용을 보관하거나 취소를 선택하여 편집 내용을 삭제하십시오.



13. 대기열에서 첫 번째 레이서를 실행하십시오.

    a. 상위 레이서 대기열 이름 옆에 있는 출발 버튼을 선택하여 각 레이서를 수동으로 출발시키십시오. 각 레이서의 차례에 실행 후 콘솔에서 10, 9, 8, 7, 6... 카운트다운 애니메이션이 나타납니다.

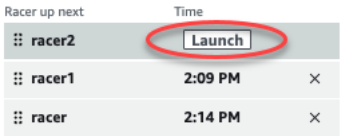
    b. Go!에서 선택한 시간 동안 모델이 실행되며 실시간으로 평가됩니다.

    c. 레이스 도중에 모델이 고장나는 경우, 레이서 대기열에 있는 별칭 옆에 있는 실행 버튼을 사용하여 레이서를 다시 시작해야 합니다.

    d. 현재 레이서가 끝나기 약 2분 전에 컨퍼런스 브리지를 통해 대기열에 있는 다음 2명의 레이서에게 연락하여 레이스를 시작할 준비가 되었는지 확인하십시오.

    e. 현재 레이서가 완주하기 30초 전에 다음 레이서에게 30초 경고를 보내십시오.

    f. 현재 레이서가 완주한 것을 확인하자마자 다음 레이서를 실행하십시오. 레이스의 종료는 콘솔에 체크무늬 깃발 아이콘으로 표시됩니다. 레이서의 속도 제어가 비활성화되고 비디오 화면에서 레이스 리플레이가 시작됩니다.



14. 시뮬레이터에 문제가 발생하는 경우 시뮬레이터 재설정을 선택할 수도 있습니다.

15. 어떤 이유로든 리더보드를 재설정하려는 경우 리더보드 순위 지우기를 선택하여 모든 항목을 지울 수도 있습니다.

16. 레이스가 끝나면 우승자 선언! 버튼을 누르고, 레이서들에게 마지막 발언을 하고, 상품이 어떻게 분배되는지 설명하고, 질문에 답하고, 화상 회의를 종료하십시오.

AWS DeepRacer League 프로덕션 플레이북을 사용하여 라이브 커뮤니티 레이스를 방송하십시오.

라이브 레이스는 지정된 날짜 및 시간에 진행되는 실시간 이벤트입니다. 이벤트 범위는 레이스 주최자 한 명이 비공개 화상 회의를 진행하는 소규모 이벤트부터 주최자, 해설자, 방송사로 구성된 작은 팀이 Twitch와 같은 라이브 스트리밍 서비스를 사용하여 공개적으로 중계하는 대규모 이벤트까지 다양합니다.

주최자 역할

다음은 AWS DeepRacer 라이브 이벤트 중에 주최자가 수행할 수 있는 권장 역할입니다. 이벤트를 계획하는 것이 복잡할수록 더 많은 도움을 요청해야 할 수도 있습니다.

주최자

레이스 주최자는 레이서를 조직하고 안내하기 위해 레이스 및 관련 화상 회의를 개최합니다. 라이브 레이스가 진행되는 동안 주최자는 주최자 컨트롤을 사용하여 레이서를 대기열에 올리고, 레이서를 실행하고, 우승자를 선정합니다. 주최자는 라이브 채널에 표시되지 않습니다.

해설자

해설자들은 레이스가 진행되는 동안 레이스에 대해 토론하며 각 이벤트의 실황, 추가 정보, 이벤트 및 참가자에 대한 내부 지식을 제공합니다. 공개 행사의 주요 연사는 해설자입니다.

방송사

방송사는 스트리밍 소프트웨어를 사용하여 미리 장면을 만들고 라이브 레이스 중에 장면을 전환합니다. 또한 방송사는 비디오 피드를 관리합니다. 방송사는 라이브 채널에 출연하지 않습니다. 이들은 이벤트 기간 동안 콘텐츠 제작자 역할을 합니다.

방송사 장면

AWS DeepRacer 이벤트의 라이브 스트림은 여러분의 레이스 스토리를 들려줍니다. 이벤트의 시작, 중간, 종료 전반에 걸쳐 참여를 장려하려면 장면을 사용하십시오. 이벤트의 여러 부분을 돋보이게 하는 그래픽 오버레이와 비디오 스트림으로 구성된 애니메이션과 레이아웃입니다.

오버레이는 레이스의 방송사 모드 창과 웹캠 스트림 또는 해설자(선택 사항) 위에 놓이는 그래픽(보통 투명한 PNG 파일)입니다. 스트리밍을 위한 마스크와 같습니다. 콘텐츠를 그 아래에 배치하여 모든 것이 매끄럽게 정렬되어 하나의 통일된 레이아웃을 만들 수 있습니다.

방송 전에 OBS와 같은 스트리밍 소프트웨어를 사용하여 장면을 설정하십시오. 이벤트가 진행되는 동안 화면을 부드럽게 넘나들며 역동적인 분위기를 만들고 시청자에게 즐거움을 선사하십시오. 예를 들어 인트로 애니메이션 장면을 사용하여 이벤트를 시작하십시오. 그런 다음 레이스 뷰와 해설자를 위한 한두 개의 창이 있는 메인 레이아웃인 PCS(Primary Content Scene)로 전환하십시오. 전체 화면 이중 해설자 또는 해설자 겸 인터뷰 대상자 장면으로 전환해 생동감을 더하고 리더보드 장면으로 마무리하십시오. 선택 사항으로, 레이스 중간에 끼어들 수 있는 상업적인 장면을 만듭니다.

AWS DeepRacer 장면 템플릿

AWS DeepRacer League 가상 서킷 팀은 라이브 커뮤니티 레이스에 사용할 수 있는 템플릿 파일 컬렉션을 만들었습니다. AWS DeepRacer 장면 템플릿을 다운로드하고 이를 사용하여 전문가 수준의 이벤트를 방송하십시오.

장면 유형 및 사용 방법

1. 인트로 AWS DeepRacer 쉴드 애니메이션
2. 콘솔 공유 전용 보기:
   - 기본 레이어 - 레이스의 방송사 모드 URL의 화면 공유입니다. 장면 프레임에 맞게 크기를 조정하십시오.
3. 단일 해설자 보기(1up):

- 기본 레이어 - 레이스의 방송사 모드 URL의 화면 공유입니다. 장면 프레임에 맞게 크기를 조정하십시오.
- 다음 레이어 - OBS Ninja 또는 로컬 웹캠(촬영 중인 해설자가 같은 방에 있는 경우) 오른쪽 상단의 PIP(Picture in Picture) 창에서 장면 프레임 아래로 가져와 크기를 조절하십시오.

4. 해설자와 인터뷰 대상자 또는 이중 해설자(2up):
   - 기본 레이어 - 레이스의 방송사 모드 URL의 화면 공유입니다. 장면 프레임에 맞게 크기를 조정하십시오.
   - 다음 레이어 - OBS Ninja 또는 로컬 웹캠(촬영 중인 해설자가 같은 방에 있는 경우) 오른쪽 상단의 PIP(Picture in Picture) 창에서 장면 프레임 아래로 가져와 크기를 조절하십시오.
   - 듀얼 웹캠 피드 또는 닌자 피드를 오른쪽 상단 창으로 가져와 크기에 맞게 조정합니다(이벤트 일주일 전에 설정 완료 - AV에서 모든 피드를 확인하고 OBS에서 카메라 할당)
5. 듀얼 해설자 전체 화면(레이싱 뷰 없음, 인터뷰만 가능):
   - 베이스 레이어 콘솔은 없고 카메라 피드는 두 개뿐입니다.
6. 종료 리더보드:
   - 장면 레이어에 리더보드 결과를 실시간으로 수동으로 입력합니다.

AWS DeepRacer 장면 템플릿 파일 팁

- 34 - 해설자를 위한 제목 구성(PIP로 이름이 포함된 사전 빌드 장면)
- 234 - 레이싱 뷰
  - 왼쪽 상단의 AWS DeepRacer League 로고를 회사 로고로 바꾸는 것을 고려해 보십시오.
  - 왼쪽 하단의 텍스트를 레이스 이름으로 바꾸고 세로 텍스트는 사용자 정보로 바꾸십시오.

라이브 개인 방송을 제작하려면 - 5분 설정

AWS DeepRacer 라이브 커뮤니티 레이스 개인 방송은 소규모 비공식 레이스에 적합합니다.

주최자 역할

- 스탠다드 레이스의 경우 주최자 한 명만 필요합니다.

하드웨어

- 권장 하드웨어 - 최소 16GB 램

- (선택 사항) 고품질 마이크, 헤드셋 또는 AirPods
- (선택 사항) LED 링 라이트: 링 라이트가 안경에 반사되지 않도록 하려면 안경을 착용자 얼굴과 비스듬히 배치하십시오.
- (선택 사항) 웹캠 및 GoPro - 영상을 다양화할 수 있습니다.

팁

- Chrome 또는 Firefox 브라우저 사용(브라우저가 최신인지 확인)
- VPN을 사용 중인 경우 연결 끊기
- 추가 탭 모두 닫기

프라이빗 라이브 AWS DeepRacer 이벤트를 실행하려면

1. AWS DeepRacer 콘솔을 엽니다.
2. 커뮤니티 레이스를 선택합니다.
3. 커뮤니티 레이스 페이지에서 레이스 만들기를 선택합니다.
4. 표준 라이브 커뮤니티 레이스를 주최할 날짜와 시간을 결정하십시오.
5. 라이브 커뮤니티 레이스를 만드는 단계를 따르기 전에 레이스 날짜 아래에서 이 기간이 가능한지 확인하십시오. 라이브 커뮤니티 레이스는 최대 4시간이 소요될 수 있습니다. 더 긴 레이스를 예약하려면 고객 지원팀에 문의하십시오.
6. 가능한 날짜가 정해지면 레이스 주최자와 참가자를 위한 해당 화상 회의를 만들어 보십시오. 청중이 거의 또는 전혀 없는 소규모 레이스를 진행하는 경우 하나의 화상 회의면 충분합니다. 대규모 개인 레이스를 개최하고 싶다면 다른 화상 회의를 만들어 청중에게 레이스를 중계하십시오.
7. 안내된 the section called “레이스 빠른 시작 만들기”의 단계에 따라 라이브 레이스를 계속 만들려면을 선택합니다.
    - 8단계에서 추천 이메일 템플릿 옆의 복사를 선택하고 레이서와 레이스 주최자를 위한 이메일을 생성할 수도 있습니다. 상품, 모델 제출 기간, 레이서들이 대기열에 등록하고 레이스를 준비하기 위해 만날 컨퍼런스 브리지 링크를 입력하십시오.
8. 레이스 당일에는 the section called “라이브 레이스를 진행해 보십시오.”의 지침을 따르십시오.
9. 상품이 있는 경우 레이스 참가자에게 배분하십시오.

라이브 공개 방송을 제작하려면 - 2시간 설정

AWS DeepRacer 라이브 커뮤니티 레이스 프리미엄 빙송은 3명 이상의 스태프가 글로벌 스트리밍 플랫폼에서 레이스를 방송하기 위해 여러 방송 장면을 사용합니다. 다음 지침에서는 Twitch를 예로 들어 설명합니다.

주최자 역할

- 주최자
- 해설자/MC
- 방송사
- Twitch 모더레이터 - 선택사항

하드웨어

- 권장 하드웨어: 최소 16GB의 RAM이 있어야 합니다.
- (선택 사항) 고품질 마이크, 헤드셋 또는 AirPods
- (선택 사항) LED 링 라이트: 링 라이트가 안경에 반사되지 않도록 하려면 안경을 착용자 얼굴과 비스듬히 배치하십시오.
- (선택 사항) 웹캠 및 GoPro: 이를 사용하여 영상을 다양화할 수 있습니다.

팁

- Chrome 또는 Firefox 브라우저를 사용하십시오(브라우저가 최신인지 확인하십시오).
- VPN을 사용 중인 경우 연결을 끊으십시오.
- 추가 탭을 모두 닫습니다.

사전 조건

- Twitch 계정 - 라이브 비디오 스트리밍 서비스.
- Twitch 스트림 키 - 소프트웨어가 비디오를 어디로 보낼지 알려줍니다.
- Open Broadcaster Software(OBS) - 비디오 녹화 및 라이브 스트리밍을 위한 무료 오픈 소스 소프트웨어입니다.
- (선택 사항) VDO Ninja(구 OBS Ninja) - 해설자와 인터뷰 대상자를 포함하도록 선택한 경우 추가 비디오 피드를 추가하고 비디오 피드 간 전환할 수 있는 도구입니다.

공개 라이브 AWS DeepRacer 이벤트를 실행하려면

1. Twitch 계정을 등록하는 방법의 단계에 따라 Twitch 계정을 설정하십시오.
2. Twitch 스트림 키를 찾으십시오. Twitch Steam 키를 찾는 방법을 알아보십시오.
3. Open Broadcaster Software(OBS)를 다운로드하십시오.
4. OBS를 사용하여 장면을 관리하는 방법을 알아보십시오. 미리 설정하십시오. 레이스를 시작하기 최소 일주일 전에 에셋을 준비하는 것이 좋습니다.

   a. 포함된 AWS DeepRacer 장면 템플릿을 다운로드하십시오.
   b. 장면을 로드하고 수정하십시오.
   c. 레이스 URL로 소스를 업데이트하십시오.
   d. 카메라를 확인하십시오.
   e. 피드에 사람들을 할당하십시오.
5. 선택적으로 해설자와 인터뷰 대상자가 방송 이벤트에 참여하는 경우 VDO Ninja(구 OBS Ninja)를 사용하여 여러 비디오 피드를 관리하십시오. OBS Ninja 사용 방법을 알아보십시오.
6. AWS DeepRacer 콘솔로 이동하여 레이스를 생성하십시오.
7. 커뮤니티 레이스를 선택합니다.
8. 커뮤니티 레이스 페이지에서 레이스 만들기를 선택합니다.
9. 공개 라이브 커뮤니티 레이스를 주최할 날짜와 시간을 결정하십시오.
10. 라이브 커뮤니티 레이스를 만드는 단계를 따르기 전에 레이스 날짜 아래에서 이 기간이 가능한지 확인하십시오. 라이브 커뮤니티 레이스의 기본 지속 시간은 4시간입니다. 더 긴 레이스를 예약하려면 고객 지원팀에 문의하십시오. 라이브 레이스가 4시간 미만인 경우에는 취해야 할 조치가 없습니다.
11. 가능한 날짜와 시간이 정해지면 레이스 주최자와 참가자를 위한 해당 화상 회의를 만드십시오.
12. 다음으로 방송사를 위한 화상 회의를 하나 더 만들어 보십시오.
13. 단계에 따라 라이브 커뮤니티 레이스를 설정하십시오.

    a. 선택 사항으로, 8단계의 레이스 설명 아래에 레이서들이 가족 및 친구들과 공유할 수 있도록 라이브 스트림 링크를 추가하십시오. 레이서를 위한 레이서 룸 컨퍼런스 브리지도 포함할 수 있습니다. 설명은 리더보드 세부 정보에 표시되어 링크에 쉽게 액세스할 수 있습니다.
    b. 선택 사항으로, 12단계에서 추천 이메일 템플릿 옆의 복사를 선택하고 레이서와 레이스 주최자를 위한 이메일을 생성할 수도 있습니다. 상품, 모델 제출 기간, 레이서들이 대기열에 등록하고 레이스를 준비하기 위해 만날 컨퍼런스 브리지 링크를 입력하십시오.
    c. 주최자 팀을 위한 다른 이메일이나 채팅을 만드십시오.

14. 레이스 당일에는 안내에 따라 the section called “라이브 레이스를 진행해 보십시오.”을 수행하십시오.
15. 우승자와 참가자를 축하하고, 상품을 배포하고, 블로그를 작성하고, 트윗하고, 글을 올리고, 퍼뜨리십시오.

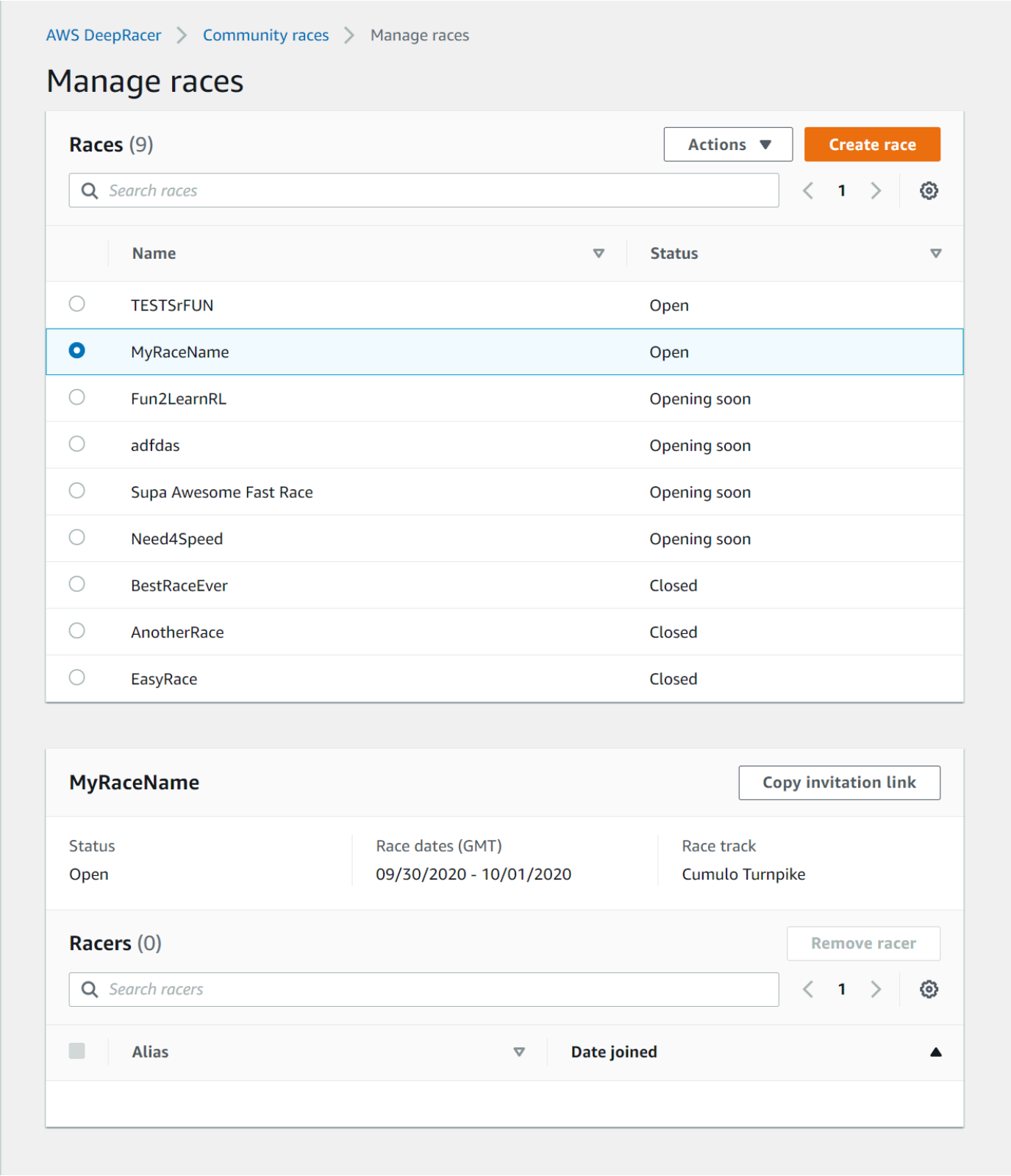
AWS DeepRacer 커뮤니티 레이스 관리

모든 커뮤니티 레이스는 비공개입니다. 초대 링크가 있는 개인만 볼 수 있습니다. 참가자는 초대 링크를 자유롭게 전달할 수 있습니다. 하지만 레이스에 참가하려면 참가자가 AWS 계정이 필요합니다. 처음 참가하는 사용자는 계정 생성 프로세스를 완료해야 레이스에 참가할 수 있습니다.

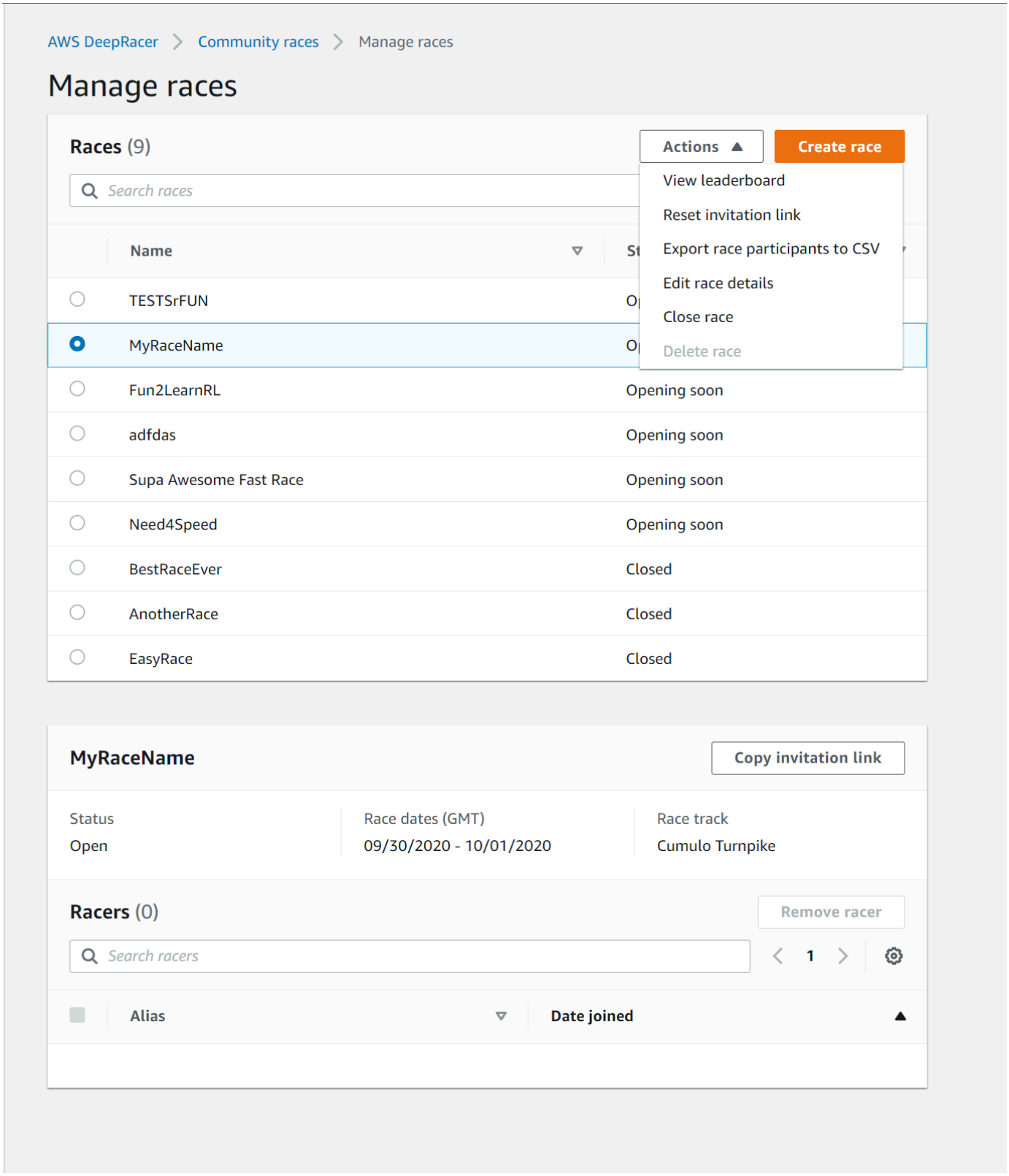
레이스 주최자는 시작일과 종료일을 포함한 레이스 세부 정보를 편집하고 참가자를 제거할 수 있습니다.

AWS DeepRacer 커뮤니티 레이스를 관리하려면

1. AWS DeepRacer 콘솔에 로그인합니다.
2. 커뮤니티 레이스를 선택합니다.
3. 레이스 관리 페이지의 레이스에서 관리하려는 레이스를 선택합니다. 참가자 목록을 포함한 선택한 레이스의 세부 정보가 표시됩니다.



4. 레이스 세부 정보를 편집하려면 작업에서 레이스 세부 정보 편집을 선택합니다.



화면의 지시에 따라 편집을 완료합니다.

5. 이벤트의 리더보드를 보려면 작업에서 리더보드 보기를 선택합니다.

6. 이벤트 초대 링크를 재설정하려면 작업에서 초대 링크 재설정을 선택합니다. 초대 링크를 초기화하면 원래 링크를 아직 선택하지 않은 사람은 레이스에 액세스할 수 없습니다. 이미 링크를 클릭하고 모델을 제출한 사용자는 모두 레이스에 남아 있습니다.

   링크를 복사하여 초대된 참가자와 공유할 수 있습니다.
7. 오픈 레이스를 종료하려면 작업에서 레이스 종료를 선택합니다. 이 작업은 지정된 종료 날짜 이전에 레이스를 즉시 종료합니다.
8. 이벤트를 삭제하려면 작업에서 레이스 삭제를 선택합니다. 이 작업은 모든 참가자의 커뮤니티 레이스에서 이 레이스와 세부 사항을 영구적으로 제거합니다.
9. 참가자를 제거하려면 한 명 이상의 레이스 참가자를 선택하고 참가자 제거를 선택한 다음 확인하여 참가자를 제거합니다.

   이벤트에서 참가자를 제거하면 해당 사용자가 레이싱 이벤트에 액세스할 수 있는 권한이 취소됩니다.

AWS DeepRacer 이벤트 주최

AWS DeepRacer 이벤트란 무엇입니까?

AWS DeepRacer는 인공 지능 및 기계 학습(AI/ML)을 사용하여 실습을 할 수 있는 재미있는 방법을 제공하는 교육 서비스입니다. AWS DeepRacer는 조직의 AI/ML 인재 격차를 해소하고 비즈니스 요구 사항에 AI/ML을 적용하는 데 도움이 될 수 있습니다.

AWS DeepRacer는 팀에 AI/ML 기술을 소개할 뿐만 아니라 팀 구성과 친선 경쟁을 장려하는 이벤트를 주최할 수 있게 해줍니다. 이 이벤트는 AWS DeepRacer 리그에서 직접 또는 가상으로 AWS DeepRacer 자동차를 경주하기 위한 강화 학습 모델을 만드는 실무 경험을 제공함으로써 기술 참가자와 비기술 참가자 모두가 기계 학습의 기초를 배울 수 있도록 도와줍니다. 또한 AWS DeepRacer 이벤트는 리더가 팀을 참여시켜 조직의 AI/ML 비전 및 목표를 달성하는 데 도움이 됩니다.

이 안내서는 AWS DeepRacer 이벤트를 계획하고 호스팅하는 데 필요한 리소스, 도구 및 예제를 제공합니다. 50명 이하의 참가자가 참여하는 AWS DeepRacer 이벤트를 계획하려면 시작하기 전에 고려할 사항으로 이동하십시오. 대규모 이벤트를 계획하는 경우(참가자가 50명 이상인 경우) AWS 계정 팀과 협력하여 이벤트를 요청하는 것이 좋습니다.

AWS DeepRacer 이벤트의 이점에 대해 자세히 알아보고 고객 사용 후기를 보려면 AWS DeepRacer 엔터프라이즈 이벤트를 참조하십시오.

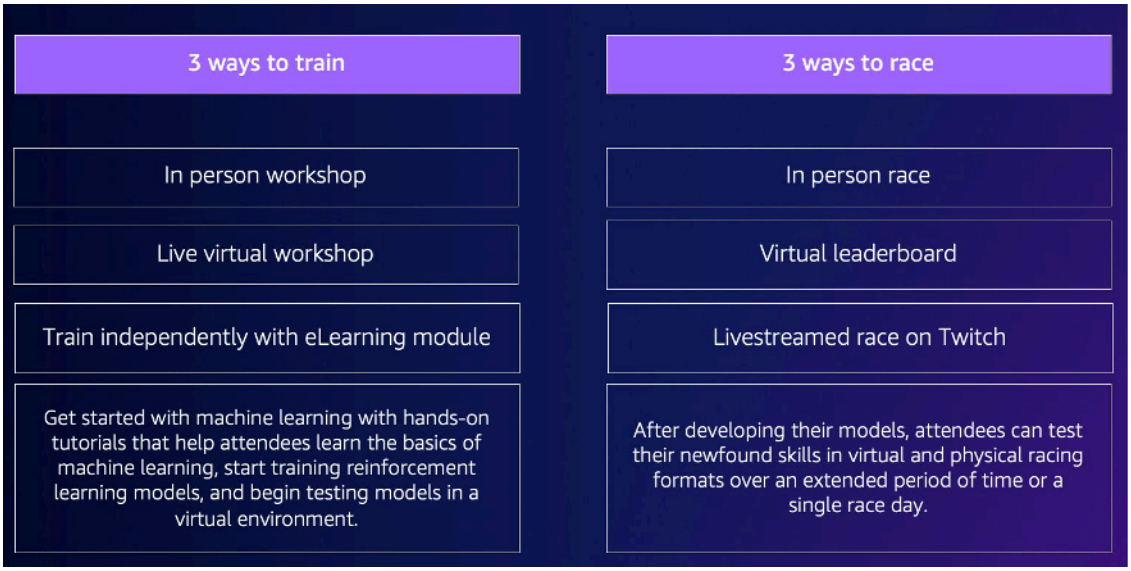
AWS DeepRacer 이벤트의 작동 방식 및 예상 사항

팀을 위한 강화 학습을 통해 훈련 및 실습을 제공하든, 새로운 인재를 유치하기 위해 조직을 홍보하든, 아니면 이 둘을 조합하든, 이 안내서는 AWS DeepRacer 이벤트를 만들고 사용자 지정하는 데 도움이 되는 도구와 리소스를 제공합니다.

AWS DeepRacer 이벤트는 사용자의 요구와 목표에 맞게 유연하지만, AWS DeepRacer 이벤트의 기본 공식에는 다음이 포함됩니다.

- 온라인 AWS DeepRacer 강화 학습 과정 또는 AWS DeepRacer Pit Crew 전문가와 함께하는 오프라인 워크숍과 같은 훈련 구성 요소.
- AWS 콘솔의 실제 모델 훈련 및 참가자가 모델을 생성하고 훈련할 수 있는 시간.
- AWS DeepRacer 레이스(예: 단일 오프라인 레이스 또는 여러 가상 레이스)를 통해 참가자는 자신들의 훈련된 기계 학습 모델이 실제로 작동하는 모습을 볼 수 있습니다. 주최할 수 있는 레이스 유형에 대한 자세한 내용은 AWS DeepRacer 레이스 유형을 참조하십시오.

- 이벤트 참가자를 위한 이벤트 후 요약 또는 다음 단계 커뮤니케이션.



AWS DeepRacer 이벤트 및 고객 하이라이트에서 기대할 수 있는 사항에 대한 자세한 내용은 AWS DeepRacer Accelerate YouTube 동영상을 참조하십시오.

시작하기 전에 고려해야 할 사항

이벤트 계획의 첫 번째 단계는 조직의 비즈니스 목적과 목표를 정의한 다음 프로젝트 계획을 수립하는 것입니다. 목표의 한 예로 “조직 내에서 재미있고 교육적인 방식으로 팀 단합을 장려하고 싶습니다.”

프로젝트 계획은 다음과 같은 의문 사항에 답할 수 있어야 합니다.

- 이벤트 리더는 누구인가요?

  조직 내에서 이 이벤트의 계획 및 실행을 주도하는 사람이 누구인지 파악하십시오. 이벤트 리더에는 시설, 인사, PR/마케팅, 경영진 후원자 또는 AWS 계정 팀과 같은 팀이 포함될 수 있습니다.
- 이벤트 소유자는 누구인가요?

  조직 내에서 이벤트 소유자를 식별하십시오. 이 사람이나 팀이 주요 주최자여야 하며 이 행사를 계획할 때 모든 결정에 참여해야 합니다.
- 이벤트 날짜는 언제인가요?

  이벤트를 언제 개최하고 싶은지 정하십시오. 참가자가 100명 이상인 대규모 행사를 고려하고 있다면 행사를 계획하고 진행하기에 충분한 리드 타임을 확보할 수 있도록 이벤트 날짜를 더 길게 잡아야 합니다. 예제 타임라인은 AWS DeepRacer 이벤트 예제를 참조하십시오.

- 예상 예산은 얼마입니까?

  이벤트 예산을 예상해 보십시오. 이벤트와 함께 계획해야 하는 두 가지 비용 고려 사항이 있습니다.

  - 이벤트 비용: 대면 이벤트를 주최하는 경우 이벤트 장소, 트랙, 디바이스 구매부터 음식, 음료, 이벤트 상품에 이르기까지 모든 비용이 포함될 수 있습니다.
  - 모델 훈련 비용:이 비용은 직원이 AWS DeepRacer 서비스를 AWS Management Console 사용하여에서 강화 학습 모델을 훈련할 때 발생합니다. 예를 들어, 각 참가자는 AWS DeepRacer 콘솔에서 하나 또는 여러 개의 모델을 학습하는 데 약 10시간이 필요합니다. 비용에 대한 자세한 내용은 AWS DeepRacer 요금을 참조하십시오.
- 이벤트에 필요한 장소는 어디입니까?

  오프라인 이벤트를 개최하는 경우 워크숍과 레이스를 위한 물리적 장소를 확보해야 합니다. 선택한 물리적 트랙 유형도 해당 위치에 맞아야 합니다.
- 예상 참가자 수는 몇 명입니까?

  이벤트에 참석하는 참가자 수를 추정해 보십시오. 참가자 수 또한 예상 예산에 영향을 미칩니다.
- 이벤트 종료 후 회고 세션을 통해 성공 여부를 가늠해 볼 계획이신가요?

  참가자가 ML 기술을 계속 발전시키고 계속 협업하도록 장려하려면 다음 단계를 알리고 참가자에게 피드백을 요청하는 것이 좋습니다. 예를 들어 참가자에게 설문조사를 보내 관심을 파악하고 조직 내 AI/ML 리더를 모집하십시오. 이를 통해 향후 AWS DeepRacer 이벤트에 누가 참여할지도 결정할 수 있습니다.
- 조직에서 이벤트를 홍보하기 위한 커뮤니케이션 전략은 무엇입니까?

  조직 내에서 이 이벤트를 알리는 것은 내부 이메일이나 캘린더 초대처럼 간단할 수 있습니다.
- 경영진 이해관계자는 누구인가요?

  행사를 후원하고 조직 내에서 사고 리더십과 협업을 장려하는 경영진을 파악하십시오.

AWS DeepRacer 레이스의 유형

프로젝트 계획의 주요 질문에 답한 후 이벤트에서 어떤 유형의 레이스를 원하는지 결정하십시오. 주최하는 레이스 유형은 예상 예산 및 이벤트 참가자 수를 기준으로 결정해야 합니다. 다음 두 가지 유형의 레이스를 주최할 수 있습니다.

가상 레이싱

가상 레이스는 여러 지역 또는 원격 팀이 가상 환경에서 협업하고 훈련된 모델을 레이싱할 수 있는 좋은 방법입니다. 가상 레이싱에는 물리적 장치와 트랙이 필요하지 않으므로 그룹 규모를 유연하게 조정할 수 있으며 위치 및 예산 제약이 있는 조직에 적합한 솔루션입니다.

가상 레이싱에는 클래식 레이싱과 라이브 레이싱의 두 가지 유형이 있습니다. 클래식 레이싱과 라이브 레이싱의 주요 차이점은 클래식 레이싱 참가자가 자신의 모델을 훈련하고 원하는 시간에 제출할 수 있다는 것입니다. 라이브 레이싱은 참가자들이 정해진 시간 내에 차례로 모델을 경주하는 것을 말합니다. 두 가지 유형의 레이스 모두 비공개 또는 공개적으로 공유할 수 있으며 리더보드를 사용하여 참가자 순위를 결정할 수 있습니다. AWS 은 Twitch에서 레이스를 방송할 수 있는 도구도 제공합니다.

가상 레이싱은 AWS DeepRacer 레이스를 시작하는 가장 빠른 방법입니다. 참가자는 콘솔에서 기계 학습 및 강화 학습에 대한 소개를 포함하는 시작하기 프로세스를 진행할 수 있습니다. 콘솔은 참가자가 첫 번째 모델을 만드는 과정을 안내합니다. 가상 레이스를 설정하는 방법에 대한 자세한 내용은 레이스 만들기 단원을 참조하십시오.

오프라인 레이싱

대면 레이스를 주최하여 팀이 직접 참석할 수 있는 매력적인 이벤트를 제공하십시오. 오프라인 레이싱에는 물리적 트랙과 AWS DeepRacer 차량 디바이스가 포함됩니다. AWS 은 트랙을 구축하고 이벤트에 사용할 차량을 조달하기 위한 옵션을 제공합니다.

AWS DeepRacer 디바이스에 대한 자세한 내용은 AWS DeepRacer 차량 작동을 참조하십시오. 물리적 트랙 및 장벽을 구성하고 주문하는 방법에 대한 자세한 내용은 물리적 트랙 구축을 참조하십시오.

이 두 가지 유형의 레이스 외에도 훈련용 온라인 훈련 또는 워크숍을 포함시키고 Twitch에서 레이스 이벤트를 생중계할 수 있습니다.

모범 사례

AWS DeepRacer 이벤트 계획의 효율성을 높이기 위해 다음 모범 사례도 권장합니다.

- 내부 리더 지원을 받으십시오. 조직 리더의 지원은 팀 참여를 장려하고 참여도와 전반적인 참가자 만족도를 높입니다.
- AWS 계정 관리자가 있는 경우 해당 관리자에게 연락하여 이벤트를 알리고 지원이 필요한지 논의합니다.
- 대규모 이벤트를 계획하는 경우(참가자가 50명 이상인 경우) AWS 계정 관리자와 조기에 자주 참여하세요. AWS 계정 팀 담당자에게 문의해야 할 사람이 확실하지 않은 경우 요청을 제출하십시오. 요청을 제출하려면 이벤트 요청을 참조하십시오. 이벤트 및 훈련에 대한 AWS DeepRacer 커뮤니티의 추가 리소스는 AWS DeepRacer Slack 채널을 참조하십시오.

- AWS DeepRacer 이벤트 목표에 맞는 예산을 수립하십시오. 예산이 계획 중인 이벤트 유형에 영향을 미칠 수 있지만, AWS DeepRacer는 비용을 제한하고 관리하는 데 도움이 되는 다중 사용자 모드와 같은 도구를 제공합니다.
- 참가자에게 상품을 제공하십시오. 팀이 전통적인 챔피언십 컵에 관심이 있든 NFT에 관심이 있든 상관없이 참여를 독려할 수 있는 상품을 마련하십시오.

AWS DeepRacer 이벤트 시작하기

조직의 목표를 정의했으면 프로젝트 계획을 사용하여 개최하려는 이벤트 유형을 좁힐 수 있습니다. 다음 예제 목표는 요구 사항과 AWS DeepRacer에서 얻고자 하는 이점을 기반으로 이벤트를 설정하는 방법을 보여줍니다.

팀 단합

소규모 그룹을 위한 팀 단합을 장려하는 일회성 지역 행사를 주최하고 싶다면 대면 또는 가상 행사를 고려해 보십시오. 이 목표를 달성하는 이벤트 유형의 예는 가상 이벤트 예시를 참조하십시오.

AI/ML 교육에 투자

기술 및 비기술 직원이 기계 학습에 더 익숙해지고 이러한 기술을 적용하기를 원한다면 경영진 지원을 더 많이 요청하는 것을 고려하고, 이벤트를 더 많은 참가자가 참여하여 조직에 더 큰 영향을 미치는 팀 간 이벤트로 만드는 방안을 고려해야 합니다. 이 이벤트에는 참가자 규모를 조정할 수 있도록 대면 이벤트와 가상 이벤트 구성 요소가 모두 포함될 수 있습니다. 이 목표를 달성하는 이벤트 유형의 예는 대면 이벤트 예시를 참조하십시오.

조직 인지도 제고 및 홍보

조직을 혁신적이고 사려 깊은 리더로 자리매김하여 더 많은 인재를 유치하고 조직 내 전반적인 인지도를 높이려면 가상 또는 오프라인 이벤트를 실시간 스트리밍하거나 사용자 지정 이벤트를 만들어 보십시오. 이 목표를 달성하는 이벤트 유형의 예는 사용자 지정 이벤트 예제를 참조하십시오.

AWS DeepRacer 이벤트 예제

다음 섹션에서는 목표 및 프로젝트 계획 요구 사항에 따라 생성할 수 있는 다양한 유형의 이벤트의 몇 가지 예를 제공합니다. 이러한 이벤트 일정은 교육 및 레이스 구성 요소에 따라 계획됩니다. 하지만 조직의 필요에 따라 이벤트 구성 요소 수에 맞게 이벤트 타임라인을 사용자 지정할 수 있습니다.

가상 이벤트 예제

가상 이벤트는 여러 위치에 있는 조직이나 여러 지역의 팀이 편리하고 비용 효율적인 방식으로 모일 수 있는 좋은 방법입니다. 가상 이벤트는 종속성이 적기 때문에 더 편리하고 비용 효율적입니다. 예를 들어 대면 이벤트처럼 물리적 트랙, 디바이스 또는 위치 공간을 고려할 필요가 없습니다. 다음 가상 이벤트 예는 타임라인 및 참가자 수와 같은 다양한 프로젝트 계획 항목에 중점을 둡니다.

	Virtual classic	Virtual live	Leaderboard + LIVE
Lead time	4 weeks	4 weeks	6-8 weeks
Cost	Organizing and submitting models in any virtual race is free. Costs incurred include: training a model, evaluating a model, and S3 storage of models.		
Recommended number of attendees	10-1000	<30	10-1000

- 2주간의 가상 클래식 레이스 이벤트:

  다음 2주간의 이벤트 일정 예시는 클래식 레이스 제출에 제한이 없고 실시간 레이스 시간 제약이 없기 때문에 참가자가 많고 레이스에 참가할 팀이 있는 이벤트를 호스팅할 수 있는 간단한 방법을 찾는 조직에 아주 좋은 옵션입니다. 모든 참가자는 레이스 기간 내에서 언제든지, 심지어 동시에 모델을 제출할 수 있습니다. 이 이벤트 예제에서는 AWS 콘솔의 커뮤니티 레이스 섹션을 사용하여 프라이빗 클래식 레이스를 생성하고 2주 후에 훈련과 레이스를 예약합니다. 커뮤니티 레이스를 설정하는 방법을 알아보려면 레이스 관리를 참조하십시오.

  - 첫째 날: 참가자는 모든 참석자를 위한 무료 온디맨드 가상 워크숍에 참석합니다. 온라인 워크숍 리소스에 대한 자세한 내용은 AWS DeepRacer: 강화 학습을 통한 주도를 참조하십시오. 또한 AWS DeepRacer Pit Crew 전문가의 도움을 받아 가상 워크숍을 진행하도록 예약할 수 있습니다.
  - 1~5일: 참가자는 AWS DeepRacer 콘솔에서 모델을 훈련, 업데이트 및 테스트합니다. 참가자는 모델을 제출하여 1~5일 동안 비공개 가상 레이스에 참가합니다. 한 AWS 계정에서 여러 참가자에 대해 AWS 콘솔을 설정하는 방법을 알아보려면 다중 사용자 모드를 참조하세요.
  - 5~10일: 5일차에 상위 10명의 우승자를 선정하고 다음 주에 새 트랙에서 열리는 새로운 비공개 가상 레이스에 참가할 수 있게 합니다.
  - 10일차: 10일차에 상위 3명의 우승자를 선정하고 레이스와 이벤트가 종료됩니다.
- 1주간의 가상 라이브 레이스 이벤트:

  다음 가상 라이브 레이스 이벤트 예시는 소규모 팀에게 레이싱의 짜릿함과 재미를 실시간으로 선사할 수 있는 좋은 방법입니다. 이러한 유형의 이벤트는 예산이 적고 모두가 실시간으로 레이스를 진

행할 수 있습니다. 라이브 레이스 진행 방법에 대한 자세한 내용은 라이브 레이스 진행을 참조하십시오.

- 첫째 날: 참가자들은 AWS DeepRacer Pit Crew 전문가와 함께하는 가상 워크숍에 참석합니다. 워크숍에 대한 자세한 내용은 AWS DeepRacer 이벤트를 참조하십시오.
- 1~5일: 참가자는 1~5일 동안 AWS DeepRacer 콘솔에서 모델을 훈련, 업데이트 및 테스트합니다.
- 5일차: 참가자들은 1~2시간 동안 온라인으로 모여 모델을 제출하고 5일째 되는 날 라이브 가상 레이스에 참가합니다.

- 가상 클래식 레이스 및 라이브 레이스 이벤트:

  다음 이벤트 예는 여러 지역의 많은 참가자를 하나로 모으려는 조직이나 여러 온라인 사무실에서 원격으로 작업하는 참가자를 하나로 모으려는 조직에 적합한 옵션입니다. 이러한 유형의 이벤트에서는 이벤트 일정이 2주에 걸쳐 진행되므로 참가자들이 모델 훈련과 레이스를 연습할 기회가 더 많아집니다. 또한 라이브 스트리밍을 시청하는 참가자들에게 이벤트가 더욱 흥미로울 수 있도록 라이브 레이스 이벤트에 아나운서를 두는 것이 좋습니다. Pro Division Finale Twitch 스트리밍을 통해 피날레 방송사들이 어떻게 레이스를 더욱 흥미진진하게 만드는지 확인해 보십시오.

  - 첫째 날: 참가자는 AWS DeepRacer Pit Crew 전문가와 함께 가상 워크숍에 참석합니다. 워크숍에 대한 자세한 내용은 AWS DeepRacer 이벤트를 참조하십시오.
  - 1~5일: 참가자는 AWS DeepRacer 콘솔에서 모델을 훈련, 업데이트 및 테스트한 다음, 모델을 제출하여 1~5일 간의 프라이빗 리그 레이스에 참가합니다.
  - 5일차: 상위 10명의 우승자를 선정하여 새 트랙에서 열리는 새로운 프라이빗 리그 레이스에 참가할 수 있는 기회를 제공합니다.
  - 5~10일 차: 다음 주에 상위 10명의 우승자를 선정하고 Twitch에서 이벤트를 생중계하며 레이스를 펼칩니다. 자세한 정보는 레이스 생중계를 참조하십시오.

오프라인 이벤트 예시

단일 물리적 공간에 참가자를 모을 수 있는 조직의 경우, AWS DeepRacer를 경험하기에 대면 이벤트보다 더 좋은 방법은 없습니다. 트랙사이드에 서서 모델이 실제 자동차에서 작동하는 모습을 보는 스릴과 흥분을 경험하는 것만큼 좋은 것은 없습니다. 일반적으로 대면 이벤트는 가상 이벤트보다 더 많은 리소스가 필요하고 비용도 더 많이 듭니다. 참가자가 50명 이상이고 필요한 예산이 있는 조직이라면 AWS DeepRacer 오프라인 경험을 활용하는 것이 좋습니다. 또한 트랙 옆에서 관람하는 참가자들을 위해 대회 결과가 실시간으로 발표되므로 레이스를 더욱 흥미진진하게 만들기 위해 이러한 대면 이벤트에 아나운서를 두는 것이 좋습니다.

	One-day workshop + race	Two-day workshop + race	Custom format
Lead time	6+ weeks	6+ weeks	10+ weeks
Recommended number of attendees	51–100	100–500	500–1,000

- 대면 1일 이벤트:

  대면 이벤트의 다음 예제는 로컬 팀에 적합하며 모든 참가자가에서 첫 번째 모델을 훈련 AWS Management Console 하고 하루 안에 경쟁할 수 있도록 합니다. 일반적인 1일 대면 이벤트는 모든 참석자가 참가할 수 있는 대면 워크숍과 레이스로 구성됩니다. 워크숍은 일반적으로 강화 학습의 기초를 설명하고 참가자에게 첫 번째 모델을 훈련할 기회를 제공하는 AWS DeepRacer Pit Crew 전문가가 주도합니다. 워크숍이 끝난 후 참가자는 모델을 제출하고 이를 AWS DeepRacer 디바이스에 업로드하여 당일 레이스를 진행할 수 있습니다.

  - 참가자는 AWS DeepRacer Pit Crew 전문가가 진행하는 AWS DeepRacer 워크숍(90~120분)에 직접 참석합니다. AWS DeepRacer 워크숍을 요청하는 방법에 대한 자세한 내용은 이벤트 요청을 참조하십시오.
  - 워크숍 이후 진행되는 오프라인 레이스(레이스당 120분 또는 약 5분).
- 이틀간의 대면 이벤트:

  다음 이틀간의 대면 이벤트 예시는 1일 이벤트와 비슷합니다. 단, 워크숍과 실제 레이스를 각각 다른 날에 진행하면 참가자가 레이스를 준비하면서 모델을 훈련하고 업데이트하는 데 더 많은 시간을 할애할 수 있다는 점이 다릅니다. 조직에서는 참가자들이 모델을 훈련하고 다듬는 데 더 많은 시간을 할애할 수 있도록 며칠 또는 몇 주 간격으로 이러한 이벤트를 주최하는 것이 일반적입니다. 워크숍과 레이스 사이에 더 많은 시간을 할애하면 참가자들이 더 경쟁적인 레이스를 할 수 있습니다.

  - 1일차: 참가자는 AWS DeepRacer Pit Crew 전문가가 진행하는 AWS DeepRacer 워크숍에 직접 참석합니다(90~120분). 오프라인 워크숍을 요청하는 방법에 대한 자세한 내용은 이벤트 요청을 참조하십시오.
  - 2일차: 참가자들은 워크숍 후 오프라인 레이스에 참가합니다(120분 또는 레이스당 약 5분).

사용자 지정 이벤트 예제

가상 이벤트와 마찬가지로 사용자 지정 이벤트는 여러 위치에 있는 팀을 위한 이벤트를 주최해야 하는 대규모 조직(참가자 100명 이상)에 적합한 옵션입니다. 사용자 지정 이벤트를 사용하면 시간과 레이스

형식에 제한이 없으므로 훈련, 워크숍, 레이스를 보다 유연하게 진행할 수 있습니다. 이러한 이벤트에는 가상 레이스와 오프라인 레이스를 모두 포함할 수 있으며, 이러한 레이스는 몇 주에 걸쳐 진행되므로 전 세계 참가자가 모델 훈련 및 협업에 더 많은 시간을 할애할 수 있습니다. 이러한 유형의 사용자 지정 이벤트는 잠재적인 물류 문제에 대비하기 위해 먼저 소규모 이벤트를 몇 개 진행해 놓는 것이 더 성공적입니다. 이러한 유형의 이벤트 또는 일련의 이벤트는 조직 내에서 기계 학습 전도사 팀을 육성하는 데도 도움이 됩니다.

- 다음 예시에서는 사용자 지정 이벤트를 3개월에 걸쳐 진행하여 여러 지역의 직원을 수용할 수 있도록 합니다.
  - 1개월 차: 여러 지역에서 열리는 글로벌 워크숍.
  - 2개월 차: 여러 지역의 참가자가 한 달 동안 진행되는 가상 리그 예선 레이스를 이용할 수 있습니다.
  - 3개월차: 오프라인 레이스 및 가상 챔피언십 컵 레이스. 이 오프라인 레이스는 실시간 스트리밍으로 글로벌 팀이 시청할 수 있습니다.

사용자 지정 이벤트를 호스팅하려면 AWS 에 문의하여 이벤트 지원을 받으세요. 이벤트 요청을 참조하십시오.

추가적인 리소스

AWS DeepRacer 이벤트와 관련된 추가 리소스는 다음 목록을 참조하십시오.

- AWS DeepRacer 블로그
- AWS DeepRacer League
- AWS DeepRacer 커뮤니티 슬랙 채널
- 기계 학습 훈련
- 기계 학습 인증
- AWS DeepRacer 훈련
- AWS DeepRacer GitHub 리포지토리
- 레이싱 팁
- AWS DeepRacer 유튜브 채널
- AWS DeepRacer 스토어

다중 사용자 모드

다중 사용자 모드 계정 설정은 조직이 한 AWS 계정에서 여러 AWS DeepRacer 참가자를 후원할 수 있는 흥미로운 방법을 제공합니다. 후원을 받는 참가자는 자신의 비용이 발생하지 않습니다. 대신 훈련 시간과 스토리지 비용은 후원 AWS 계정으로 청구됩니다. 다중 사용자 모드 계정 설정을 통해 AWS DeepRacer 이벤트 주최자는 개별 참가자, 그룹 또는 모든 참가자의 교육 시간 및 모델에 대한 기본 할당량을 업데이트하여 예산을 설정하고 지출을 모니터링 및 제어할 수 있습니다.

다음 섹션에서는 관리자 또는 참가자로서 AWS DeepRacer 다중 사용자 모드를 사용하는 방법을 설명합니다.

> **Note**
>
> 계정 후원이 가능한 다중 사용자 모드는 AWS DeepRacer 서비스에서만 사용할 수 있습니다.

주제

- 다중 사용자 모드 설정(관리자)
- AWS DeepRacer 다중 사용자 경험 (참가자)

다중 사용자 모드 설정(관리자)

다중 사용자 계정 설정을 통해 주최자(계정 관리자 등)는 참가자에게 주최자 계정 ID로 AWS DeepRacer 서비스에 대한 액세스 권한을 제공할 수 있습니다. 또한 참가자의 훈련 시간에 대한 사용량 할당량을 설정하고, 훈련 및 스토리지에 대한 지출을 모니터링하고, 훈련을 시작 및 중단하고, AWS DeepRacer 콘솔에서 계정의 모든 사용자에 대한 모델을 보고 관리할 수 있습니다.

다중 사용자 모드는 개별 AWS 계정이 없는 여러 참가자가 있는 대규모 이벤트에 특히 유용합니다. AWS DeepRacer 관리자는 이벤트의 각 참가자에 대한 계정을 생성하고 관리하는 대신 단일 AWS 계정을 통해 모든 후원 참가자를 호스팅할 수 있습니다.

다중 사용자 모드에서는 후원 참가자가 자신의 비용을 들이지 않고도 경쟁하고 훈련할 수 있습니다. 훈련 및 보관 비용은 후원하는 다중 사용자 AWS 계정에 청구됩니다. 관리자가 참가자의 사용을 후원하는 것을 중단해도 참가자는 레이서 별칭과 프로필을 유지합니다.

다중 사용자 이해관계자

이 안내에서는 다중 사용자 모드 설정 및 사용에 대한 다음과 같은 일반적인 다중 사용자 이해 관계자를 참조합니다.

- AWS IAM/SSO 구성을 위한 관리자입니다. IAM/SSO 구성 AWS 관리자는 AWS DeepRacer 관리자 및 참가자가 다중 사용자 모드를 사용할 수 있도록 IAM 또는 SSO를 설정합니다. IAM/SSO AWS 의 관리자는 IAM 및 SSO 관리자 권한을 가집니다. IAM 사용자 생성에 대한 자세한 정보는 AWS 계정에서 IAM 사용자 생성 섹션을 참조하십시오.
- AWS DeepRacer 관리자. AWS DeepRacer 관리자는 AWS DeepRacer 참가자의 후원을 관리하며 후원을 일시 중지 및 재개하고, 모델 및 아티팩트를 삭제하고, 가상 레이스를 구성 및 호스팅하고, 다중 사용자 모드를 활성화 및 비활성화할 수 있습니다. AWS DeepRacer 관리자는 AWSDeepRacerAccountAdminAccess 권한을 가집니다.
- AWS DeepRacer 참가자. AWS DeepRacer 참가자는 다중 사용자 모드에서 관리자 AWS 계정으로 이벤트에 참여하도록 초대됩니다. 참가자는 후원자 계정에서 모델을 훈련, 평가 및 저장할 수 있는 AWSDeepRacerDefaultMultiUserAccess 권한을가집니다. 참가자는 또한 레이서 프로필을 구성하고, 가상 레이스에 참가하고, 모델을 다운로드하여 실제 AWS DeepRacer 차량에 배포할 수 있습니다.

이 안내서에서는 다음을 수행합니다.

- 1단계. 필수 조건 수행
- 2단계. AWS DeepRacer 계정에서 다중 사용자 모드를 활성화합니다.
- 3단계. 참가자를 초대하십시오.
- 4단계. 사용량 할당량을 설정하십시오.
- 5단계. 후원 참가자의 사용량을 모니터링하십시오.

1단계. AWS DeepRacer 다중 사용자 모드의 필수 조건

다중 사용자 모드에 대한 다음 필수 조건을 완료합니다.

- 다중 사용자를 위한 AWS DeepRacer 관리자 권한으로 계정을 설정하십시오. 다중 사용자 모드로 레이스를 조직하고 일반적인 AWS DeepRacer 관리자 작업을 수행하는 경우, AWSDeepRacerAccountAdminAccess 권한을 가진 AWS DeepRacer 관리자로 계정을 설정해야 합니다.
- 후원하려는 참가자에게 AWS 콘솔 액세스 및 레이서 정책 권한을 제공합니다.

다중 사용자를 위한 AWS DeepRacer 관리자 권한으로 계정을 설정하십시오.

다중 사용자 모드를 위한 AWS DeepRacer 관리자로 설정하려면 사용자, 그룹 또는 역할에 연결된 IAM AWS DeepRacer 관리자 정책인 AWSDeepRacerAccountAdminAccess가 있어야 합니다. 조직에 따라 콘솔을 사용하여 사용자 또는 역할을 생성하고 필요한 IAM 정책을 첨부하여 관리자 정책을 직접 설정하거나 IT 관리자에게 제공하도록 할 수 있습니다. 필요한 관리자 정책에 대한 자세한 내용은 AWSDeepRacerAccountAdminAccess를 참조하십시오. IAM 정책에 대한 자세한 내용은 IAM 사용 설명서의 액세스 관리 섹션을 참조하십시오.

후원 참가자에게 AWS 콘솔 액세스 권한 제공

후원하는 레이서에게 AWS DeepRacer 콘솔에 대한 액세스 권한을 제공하려면 AWS IAM Identity Center 또는와 같은 표준 AWS 권한 부여 프로토콜을 사용하는 것이 좋습니다AWS Identity and Access Management. 조직의 기존 SSO를 통해 액세스를 제공할 수도 있습니다. 참가자가 제공한 자격 증명을 사용하여 AWS DeepRacer 콘솔에 로그인하면 AWS 플레이어 계정을 생성하여 로그인하고 AWS 계정에서 AWS DeepRacer 콘솔에 액세스하라는 메시지가 표시됩니다. AWS 플레이어 계정에 대한 자세한 내용은 AWS 플레이어 계정을 참조하십시오.

IAM을 사용하여 후원 참가자에게 AWS 콘솔 액세스 제공

1. 각 참가자의 IAM 사용자 이름과 비밀번호를 생성합니다. AWS 계정의 IAM 사용자 생성을 참조하십시오.
2. 각 참가자에게 AWS DeepRacerDefaultMultiUserAccess 권한을 부여하십시오. 자세한 내용은 AWS DeepRacer에 대한AWS 관리형 정책을 참조하십시오.
3. 참가자에게 IAM 사용자 이름과 암호, 콘솔 링크를 이메일로 보내십시오. 제공된 링크를 사용하여 IAM 사용자 이름과 암호를 입력하면 참가자가 콘솔에 액세스할 수 있습니다. IAM 사용자 생성에 대한 자세한 정보는 AWS 계정에서 IAM 사용자 생성 섹션을 참조하십시오.

IAM Identity Center를 사용하여 후원 참가자에게 AWS 콘솔 액세스 권한 제공

1. https://console.aws.amazon.com/singlesignon/에서 IAM Identity Center 콘솔을 열고 사용자 지정 권한 집합을 만들고 계정에 사용자를 할당합니다. 자세한 내용은 권한 집합을 참조하십시오.
2. 사용자 지정 권한 집합을 생성할 때 다음 값을 제공하십시오.

- 릴레이 상태: `https://console.aws.amazon.com/deepracer/home?region=us-east-1#getStarted`

> **Note**
>
> 릴레이 상태는 계정 내 참가자를 지정된 URL로 리디렉션합니다. 이 경우 참가자를 AWS DeepRacer 콘솔로 안내합니다.

- AWS 관리형 정책: `AWS DeepRacerDefaultMultiUserAccess`

필수 조건을 충족했으면 이제 다중 사용자 모드를 활성화하고 참가자를 초대하여 계정을 통해 레이스를 진행할 수 있습니다.

2단계: 다중 사용자 계정 모드 활성화

AWS DeepRacer 관리자 계정을 설정하고 후원 참가자에게 콘솔 액세스 및 권한을 부여한 후에는 AWS DeepRacer 계정에서 다중 사용자 모드를 활성화할 수 있습니다.

> **Note**
>
> 기본적으로 다중 사용자 모드의 참가자를 후원하는 계정에는 할당량이 있습니다. 자세한 내용은 사용량 모니터링의 계정 할당량에 대한 섹션을 참조하십시오.

1. 왼쪽 탐색 창에서 다중 사용자 관리 및 설정 페이지로 이동합니다.
2. 다중 사용자 계정 활성화 모드에서 다중 사용자 모드 활성화를 켜십시오.
3. 다중 사용자 모드 활성화 대화 상자에서 확인란을 선택하여 후원 참가자에게 필요한 액세스 및 권한이 있는지 확인합니다.
4. 다중 사용자 모드 활성화를 선택합니다.

   사전 조건을 충족하고 다중 사용자 모드를 활성화하면 후원을 받는 각 참가자가 모든 훈련 및 스토리지 요금이 관리자 AWS 계정으로 청구되는 레이스를 만들고 모델을 훈련할 수 있습니다. 기본적으로 참가자는 3개의 동시 진행 모델 할당량을 가지며 한 번에 최대 10개의 오픈 레이스 또는 향후 레이스를 관리할 수 있습니다(라이브, 클래식 및 학생 레이스 포함).

다중 사용자 계정 모드 비활성화

다중 사용자 모드를 비활성화하면 관리자 계정으로 새 프로필을 만들 수 없고 이전에 후원한 참가자의 프로필이 관리자 계정에 더 이상 표시되지 않습니다. 참가자는 더 이상 AWS 플레이어 계정에 로그인하라는 메시지가 표시되지 않으며 관리자 계정으로 생성된 모델에 액세스하거나 훈련할 수 없습니다.

관리자는 후원 참가자의 모델을 다운로드, 저장 및 가져올 수 있습니다.

1. 다중 사용자 관리 및 설정 페이지로 이동합니다.
2. 다중 사용자 계정 모드 비활성화에서 다중 사용자 모드 비활성화를 선택합니다.
3. 다중 사용자 모드 비활성화 대화 상자에서 확인란을 선택하여 다중 사용자 모드를 비활성화할지 확인합니다. 다중 사용자 모드 비활성화를 선택합니다.

   다중 사용자 모드가 비활성화되었습니다.

> **Note**
>
> 후원하는 AWS 다중 사용자 계정으로 생성된 모든 모델은 유지되며 모델이 삭제될 때까지 해당 AWS 계정에서 모델 스토리지 비용이 계속 발생합니다.

3단계: 후원을 받을 참가자 초대

제공된 이메일 템플릿을 사용하여 참가자를 초대하여 후원 참가자로 훈련하고 레이스할 수 있습니다.

참가자 초대하기

1. 왼쪽 탐색 창에서 다중 사용자 관리 및 설정 페이지로 이동합니다. 사용자 초대 섹션의 다중 사용자 모드 설정에서 초대 템플릿 보기를 선택합니다.
2. 이메일 클라이언트 애플리케이션에 표시되는 이메일 템플릿을 복사한 다음 이를 사용하여 후원를 받도록 초대하려는 참가자에게 보낼 이메일을 만드십시오. 회사의 기존 SSO를 사용하는 경우 참가자가 사용할 SSO URL을 포함할 수 있습니다. 또는 참가자가 AWS 콘솔에 로그인하는 데 사용할 IAM 자격 증명을 제공할 수 있습니다.

4단계: 사용량 할당량 설정

후원 참가자가 초대 이메일을 받고 사용자의 계정으로 프로필을 만들면 사용량 모니터링 화면의 후원 사용자 목록에 해당 참가자가 나타납니다. 그런 다음 이 화면에서 후원 참가자의 이용 가능한 훈련 시

간 및 모델에 대한 사용량 할당량을 설정할 수 있습니다. 할당량을 설정하면 계정 내 참가자당 비용을 관리하고 참가자가 사용량 할당량을 초과하지 않도록 할 수 있습니다. 또한 필요에 따라 사용량 할당량을 늘리거나 줄여 후원 참가자에게 AWS DeepRacer 모델을 효과적으로 훈련하는 데 필요한 시간을 제공할 수 있습니다.

> **Note**
>
> 다중 사용자 모드의 후원 참가자는 기본적으로 5시간의 훈련을 받습니다.

후원 레이서의 사용량 할당량을 편집하려면

1. 왼쪽 탐색 창에서 다중 사용자 관리 및 사용량 모니터링 화면으로 이동합니다. 후원 사용자의 사용량 모니터링 화면에서 할당량을 설정하려는 참가자를 선택합니다. 작업을 선택하여 드롭다운 목록을 열고 사용량 할당량 설정을 선택합니다.
2. 사용량 할당량 설정 팝업에 선택한 참가자의 최대 훈련 시간 및 최대 모델 수를 입력합니다. 변경 내용을 저장하려면 확인을 선택하고 변경 내용을 취소하려면 취소를 선택합니다.

5단계: 사용량 모니터링

예상 지출 및 모델 훈련 시간을 포함하여 후원 참가자의 사용량을 모니터링할 수 있습니다. 또한 참가자의 후원을 일시 중지하고, 모델을 삭제하고, 사용 요약을 볼 수 있습니다. AWS DeepRacer 다중 사용자 관리에서 사용량 모니터링과 관련된 모든 작업은 사용량 모니터링 페이지에서 수행합니다.

후원 레이서의 경비에 대한 모든 정보는 추정치일 뿐이며 예산 책정이나 비용 계산 목적으로 사용해서는 안 됩니다. 견적은 미국 달러 기준이며 특별 가격은 반영되지 않습니다. 요금에 대한 자세한 내용은 요금을 참조하십시오.

다중 사용자 모드의 계정 할당량

기본적으로 다중 사용자 모드의 후원 계정에는 모든 후원 프로필에서 공유되는 할당량은 다음과 같습니다.

- 동시 훈련 작업 100개
- 100개의 동시 평가 작업
- 오픈 레이스 또는 미래 레이스 100회 (라이브, 클래식, 학생 레이스 포함)
- 자동차 1,000대

- 프라이빗 리더보드 50개

할당량을 조정하려면 고객 서비스에 문의하십시오.

예상 지출을 보려면

사용량 모니터링 페이지의 사용량 모니터링에서 참가자의 사용량에대한 예상 요약을 볼 수 있습니다.

결제 알림을 설정하려면

계정에 결제 알림을 설정할 수 있습니다. 결제 알림을 통해 지출에 대한 최신 정보를 확인할 수 있습니다. 자세한 내용은 결제 단원을 참조하십시오.

후원 일시 중지

단일 참가자, 여러 참가자 또는 모든 참가자의 후원을 일시 중지할 수 있습니다. 후원을 일시 중지하면 후원 참가자가 사용자의 계정에서 새 모델을 만들거나 모델을 훈련할 수 없습니다. 진행 중인 훈련은 완료될 때까지 진행되며 예상 지출액에 포함됩니다. 언제든 후원을 재개 할 수 있습니다. 다중 사용자 액세스가 일시 중지된 참가자는 여전히 모델을 보고 리더보드에 모델을 게시할 수 있지만 비용이 발생하는 활동은 할 수 없습니다.

1. 사용량 모니터링 페이지의 사용량 모니터링에 있는 후원 사용자 섹션에서 후원을 일시 중지하려는 사용자를 선택합니다.
2. 후원 일시 중지를 선택합니다.
3. 후원 일시 중지 대화 상자에서 후원 일시 중지를 선택하여 후원을 일시 중지합니다. 후원을 일시 중지하지 않으려면 취소를 선택하십시오.

후원을 재개하려면

후원을 일시 중지한 레이서의 후원을 재개할 수 있습니다.

1. 사용량 모니터링 페이지의 사용량 모니터링에 있는 후원 사용자 섹션에서 후원을 재개하려는 레이서를 선택합니다.
2. 후원 재개를 선택합니다.

레이서들의 모델을 보려면

- 내 모델 페이지의 모델 아래에서 내 모델과 사용자 모델을 볼 수 있습니다.

다음 단계

다중 사용자 모드를 설정하고 활성화한 후 다음 단계를 수행하십시오.

- 커뮤니티 레이스 만들기
- AWS DeepRacer 워크숍을 요청하십시오.

커뮤니티 레이스 만들기

커뮤니티 레이스는 후원 참가자가 강화 학습을 경험할 수 있는 흥미로운 방법을 제공합니다.

커뮤니티 레이스를 만들고 후원 참가자를 초대할 수 있습니다.

자세한 내용은 the section called “레이스 빠른 시작 만들기” 섹션을 참조하십시오.

워크숍 요청

60분 온라인 또는 오프라인 워크숍을 통해 AWS DeepRacer에 대해 자세히 알아보려면 워크숍을 요청할 수 있습니다.

자세한 내용은 워크숍을 참조하십시오.

AWS DeepRacer 다중 사용자 경험 (참가자)

이 안내에서는 다중 사용자 모드에서 조직의 계정을 통해 프로필을 후원받는 개별 참가자의 경험을 보여줍니다.

AWS DeepRacer는 AWS DeepRacer 모델을 교육하고 레이싱하여 강화형 기계 학습(RL)을 경험할 수 있는 흥미로운 방법을 제공합니다. 조직은 프로필이 자신의 AWS 계정으로 후원되도록 할 수 있는 기회를 제공할 수 있습니다. 모델 훈련, 평가 및 저장을 포함하여 생성하는 모든 요금은 로그인에 사용한 AWS 계정으로 청구됩니다. 프로필을 후원하는 AWS 계정의 관리자는 모델, 자동차 및 리더보드를 보고, 훈련 시간을 일시 중지하고, 훈련 시간과 스토리지 할당량을 조정하고, 프로필 후원을 중지할 수 있습니다.

스폰서 레이서 가입 절차의 일환으로 AWS 플레이어 계정을 생성합니다. 계정은 사용자가 보관하며 여러 다른 AWS 서비스와 함께 사용할 수 있는 휴대용 프로필입니다. 자세한 내용을 알아보려면 AWS 플레이어 계정을 참조하십시오.

사전 조건

조직의 이벤트 코디네이터가 AWS 콘솔의 로그인 자격 증명이 포함된 AWS DeepRacer 가입 초대를 공유합니다. 이 자격 증명을 사용하여 콘솔에 로그인합니다. 또한 설정의 일부로 레이서 프로필과 AWS 플레이어 계정을 생성합니다.

이 안내에서는 다음 단계를 다룹니다.

- 후원 계정의 자격 증명을 사용하여 AWS 콘솔에 로그인합니다.
- AWS 플레이어 계정을 생성하거나 로그인합니다.
- 프로필을 사용자 지정하십시오.
- 모델을 훈련합니다.
- 후원 사용량을 확인하십시오.
- (선택 사항) 추가 후원 시간을 요청하십시오.

1단계. 후원 계정의 자격 증명을 사용하여 AWS 콘솔에 로그인합니다.

후원 참가자로 AWS DeepRacer를 시작하려면 이벤트 코디네이터로부터 받은 초대장에 제공된 자격 증명을 사용하여 콘솔에 로그인합니다.

AWS 콘솔에 후원 참가자로 로그인하려면

1. 이벤트 코디네이터로부터 받은 초대장에 제공된 자격 증명을 사용하십시오.
2. 콘솔에서 로 이동합니다 AWS DeepRacer.

   AWS 플레이어 계정 페이지가 나타납니다.

2단계. AWS 플레이어 계정 생성 또는 로그인

1. AWS 플레이어 계정 페이지에서 플레이어 계정을 만들거나 기존 AWS 플레이어 계정으로 로그인합니다.
   - 아직 계정이 없는 경우 계정 생성을 선택하고 이메일 주소와 비밀번호를 입력한 다음 계정 생성을 선택합니다.
   - AWS 플레이어 계정이 이미 있는 경우 이메일과 암호를 입력하고 로그인을 선택합니다.
2. 계정 설정을 확인하기 위한 메시지가 지정한 이메일 주소로 전송됩니다.

3. 확인 코드 상자에 이메일로 받은 코드를 입력하고 등록 확인을 선택합니다.

> **Note**
>
> 확인 코드를 입력할 때까지 현재 페이지에 머무르십시오.

이제 AWS DeepRacer 콘솔에 후원 참가자로 로그인했습니다.

4. 3단계로 진행하여 레이서 프로필을 사용자 지정합니다.

3단계. 프로필 사용자 지정하기

프로필 이미지를 편집하고 레이서 이름을 추가하여 프로필을 사용자 지정합니다. 언제든지 레이서 프로필을 업데이트하고 변경할 수 있습니다. 또한 거주 국가와 연락처 이메일을 추가하여 AWS DeepRacer 리그에서 획득한 상금에 대한 소식을 받을 수 있습니다. 또한 AWS DeepRacer 리그에서 성과에 대한 업적을 받은 경우 내 레이서 프로필 페이지에서 소셜 미디어에 이를 공유할 수있습니다.

> **Note**
>
> AWS DeepRacer 리그 레이싱 이벤트 및 모델 훈련에 참여하려면 레이서 이름을 생성하고 거주 국가를 추가해야 합니다. 레이서 이름은 전역적으로 고유해야 합니다. 거주 국가를 선택하면 해당 국가는 레이싱 시즌 동안 고정됩니다.

레이서 프로필 이미지를 사용자 지정하려면

1. 왼쪽 탐색 창에서 내 레이서 프로필 페이지로 이동합니다.
2. 내 레이서 프로필 페이지에서 편집을 선택합니다.
3. 내 레이서 프로필 대화 상자의 드롭다운 목록에서 항목을 선택하여 레이서 프로필 이미지를 사용자 지정합니다.
4. 저장(Save)을 선택합니다.

레이서 이름을 사용자 지정하려면

1. 왼쪽 탐색 창에서 내 레이서 프로필 페이지로 이동합니다.
2. 내 레이서 프로필 페이지에서 편집을 선택합니다.

3. 내 레이서 프로필 대화 상자에서 레이서 이름 변경을 선택하고 프로필 이름을 입력합니다.
4. 저장(Save)을 선택합니다.

4단계. 모델 훈련

프로필을 사용자 지정했으면 모델 훈련을 시작할 준비가 된 것입니다. 자세한 내용은 AWS DeepRacer 모델 훈련 및 평가 섹션을 참조하십시오.

5단계. 후원 사용량 보기

후원 시간과 모델을 추적하여 최대한으로 활용할 수 있습니다.

후원 시간 사용량 및 저장된 모델을 보려면

- 내 레이서 프로필 페이지에서 총 사용 시간 및 저장된 모델 수에 대한 스폰서 사용량을 확인하십시오.

6단계. (선택 사항) 추가 후원 시간 요청

후원 참가자는 5시간의 무료 교육 시간을 받게 됩니다. 무료 후원 시간이 부족한 경우 계정 관리자 또는 이벤트 주최자에게 추가 시간을 요청할 수 있습니다. 또는 추가 후원 시간을 이용할 수 없는 경우 자체 AWS DeepRacer 계정을 생성하여 AWS DeepRacer 여정을 계속할 수 있습니다. 훈련 및 스토리지 비용에 대한 자세한 내용은 요금을 참조하십시오.

AWS DeepRacer Student에 사용되는 교육자 도구

이 섹션에서는 AWS DeepRacer 기계 학습 커리큘럼을 강의실에 통합하고, AWS DeepRacer Student 실습 랩을 개최하고, 학생 커뮤니티 레이스를 만드는 데 필요한 정보와 리소스를 제공합니다.

AWS DeepRacer Student를 강의실에 통합하십시오

AWS DeepRacer를 이제 처음 시작하는 교육자라면 AWS DeepRacer Student 교육자 플레이북을 읽어 보는 것이 좋습니다.

커리큘럼 플레이북

AWS DeepRacer Student 커리큘럼 플레이북에는 각 AWS DeepRacer Student 모듈의 개요, 학습 목표, 학습 결과, 주요 개념, 지원 자료, 평가 및 활동 제안 사항이 요약되어 있습니다.

학생 실습 플레이북

AWS DeepRacer Student 실습 플레이북은 교육자가 AWS DeepRacer Student 실습을 진행하는 데 필요한 정보와 리소스를 제공합니다. 실습은 AWS DeepRacer Student 리그 레이스, 프라이빗 커뮤니티 레이스, 라이브 가상 레이싱과 같은 가상 이벤트 및 실제 트랙과 AWS DeepRacer 디바이스를 사용한 오프라인 이벤트로 구성됩니다.

학생 커뮤니티 레이스 생성

교육자 플레이북을 시작한 후에는 AWS DeepRacer 콘솔의 커뮤니티 레이스를 사용하여 AWS DeepRacer Student 리그에서 학생들을 위한 레이스를 생성합니다. 레이스 초대 링크를 공유하여 학생 레이스 참가자를 초대합니다.

에듀케이터는 AWS DeepRacer 콘솔에 로그인하여 레이스를 만들고 조직하는 데 AWS 계정이 필요하지만, 학생은 AWS DeepRacer Student League에 로그인하고, 프로필을 업데이트하고, 무료 과정을 수강하고, AWS DeepRacer 모델을 생성하는 데 이메일 주소만 있으면 됩니다. 또한 교육자는 이메일 주소를 사용하여 AWS DeepRacer Student 리그에 계정을 생성하여 커리큘럼을 미리 보고, 레이스 경험을 시험해보고, 학생들의 진행 상황을 모니터링할 수 있습니다.

다음 주제 중 하나를 진행하여 AWS DeepRacer Student 가상 레이스를 만들거나 관리합니다.

주제

- the section called “학생 레이스 생성”
- the section called “학생 레이스를 사용자 지정합니다.”

- the section called “학생 레이스 관리”

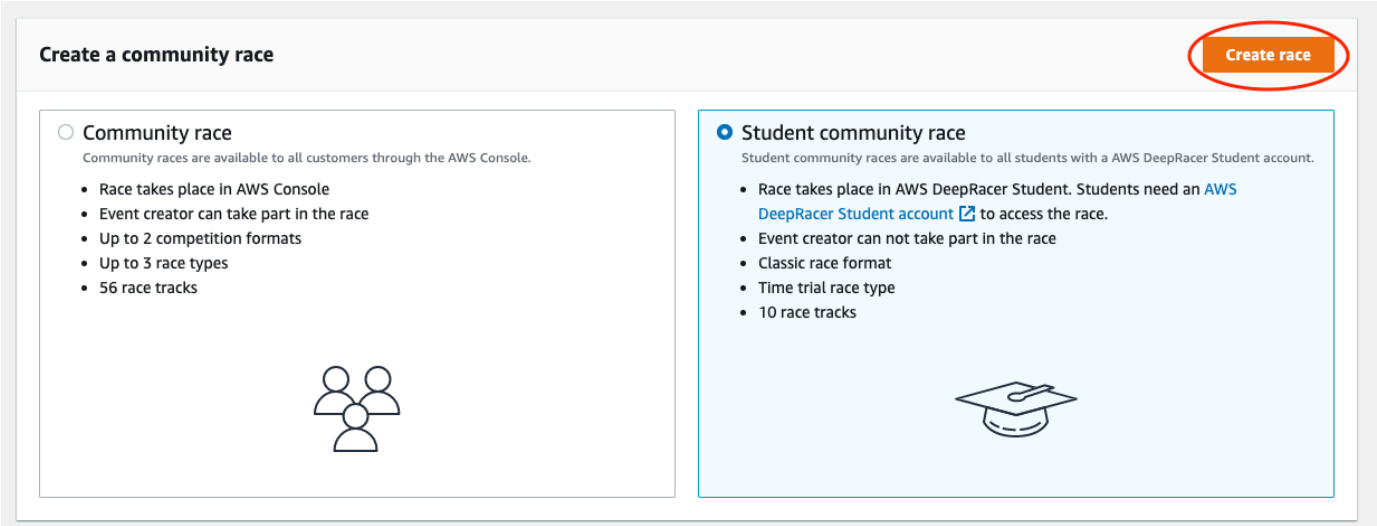
AWS DeepRacer Student 커뮤니티 레이스 생성

기본 학생 커뮤니티 레이스 설정을 사용하여 가상 레이스를 빠르게 설정할 수 있습니다.

학생 커뮤니티 레이스는 실시간 상호 작용이 필요하지 않은 비동기 이벤트입니다. 참가자는 초대 링크를 받아야 레이스에 모델을 제출하고 리더보드를 확인할 수 있습니다. 레이서는 날짜 범위 내에서 언제든지 모델을 무제한으로 제출하여 리더보드에 오를 수 있습니다. 제출된 모델의 클래식 레이스 결과 및 동영상은 레이스가 시작되자마자 리더보드 페이지에서 볼 수 있습니다.

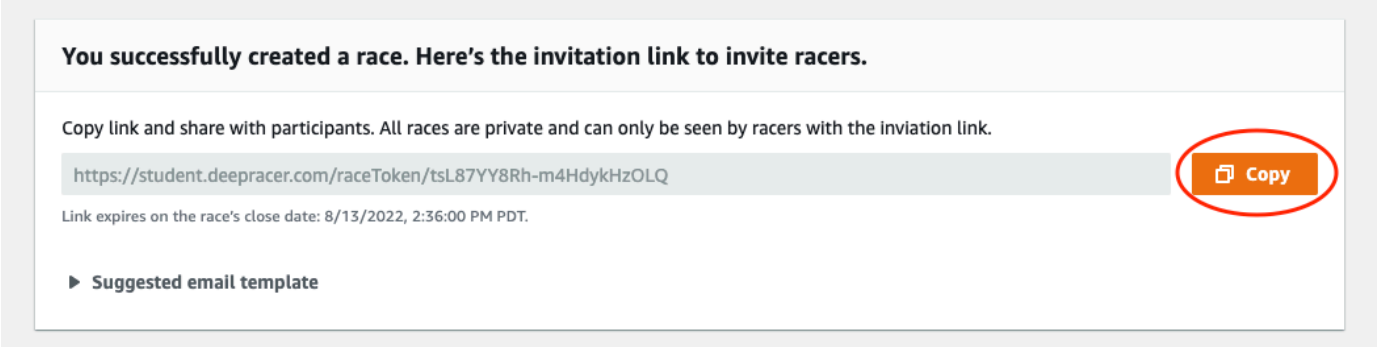
학생 커뮤니티 레이스 만들기 시작

1. AWS DeepRacer 콘솔을 엽니다.
2. 커뮤니티 레이스 페이지에서 학생 커뮤니티 레이스를 선택합니다.
3. 규칙 생성을 선택합니다.



4. 레이스에 대한 설명이 포함된 원본 이름을 입력합니다.
5. 이벤트 시작 날짜 및 시간을 24시간 형식으로 지정합니다. AWS DeepRacer 콘솔은 사용자의 시간대를 자동으로 인식합니다. 종료 날짜 및 시간도 입력합니다.
6. 다음을 선택하여 기본 설정을 사용합니다. 모든 옵션에 대해 알아볼 준비가 되면 the section called “학생 레이스를 사용자 지정합니다.”로 이동합니다.
7. 레이스 세부 정보 검토 페이지에서 레이스 사양을 확인하십시오. 변경하려면 편집 또는 이전을 선택하여 레이스 세부 정보 페이지로 돌아가십시오. 초대 링크를 받을 준비가 되면 제출을 선택합니다.

8. 레이스를 공유하려면 링크를 복사하여 추천 이메일 템플릿, 문자 메시지, 자주 사용하는 소셜 미디어 애플리케이션에 붙여 넣으십시오. 모든 레이스는 초대 링크가 있는 레이서만 볼 수 있습니다. 링크는 레이스 마감일에 만료됩니다.



9. 학생 레이스 일정이 종료되면 레이스 관리 페이지의 레이서 아래에 누가 모델에 참가시켰는지, 그리고 누가 아직 모델을 등록해야 하는지 기록해 두십시오.

레이스 관리를 선택하여 선택한 레이스 트랙 변경, 레이스 설명 추가, 순위 지정 방법 선택, 레이서 재설정 횟수 정의, RL 모델이 레이스에 출전하기 위해 완료해야 하는 최소 랩 수 정의, 오프 트랙 페널티 설정, 기타 레이스 세부 정보 사용자 지정을 진행합니다.

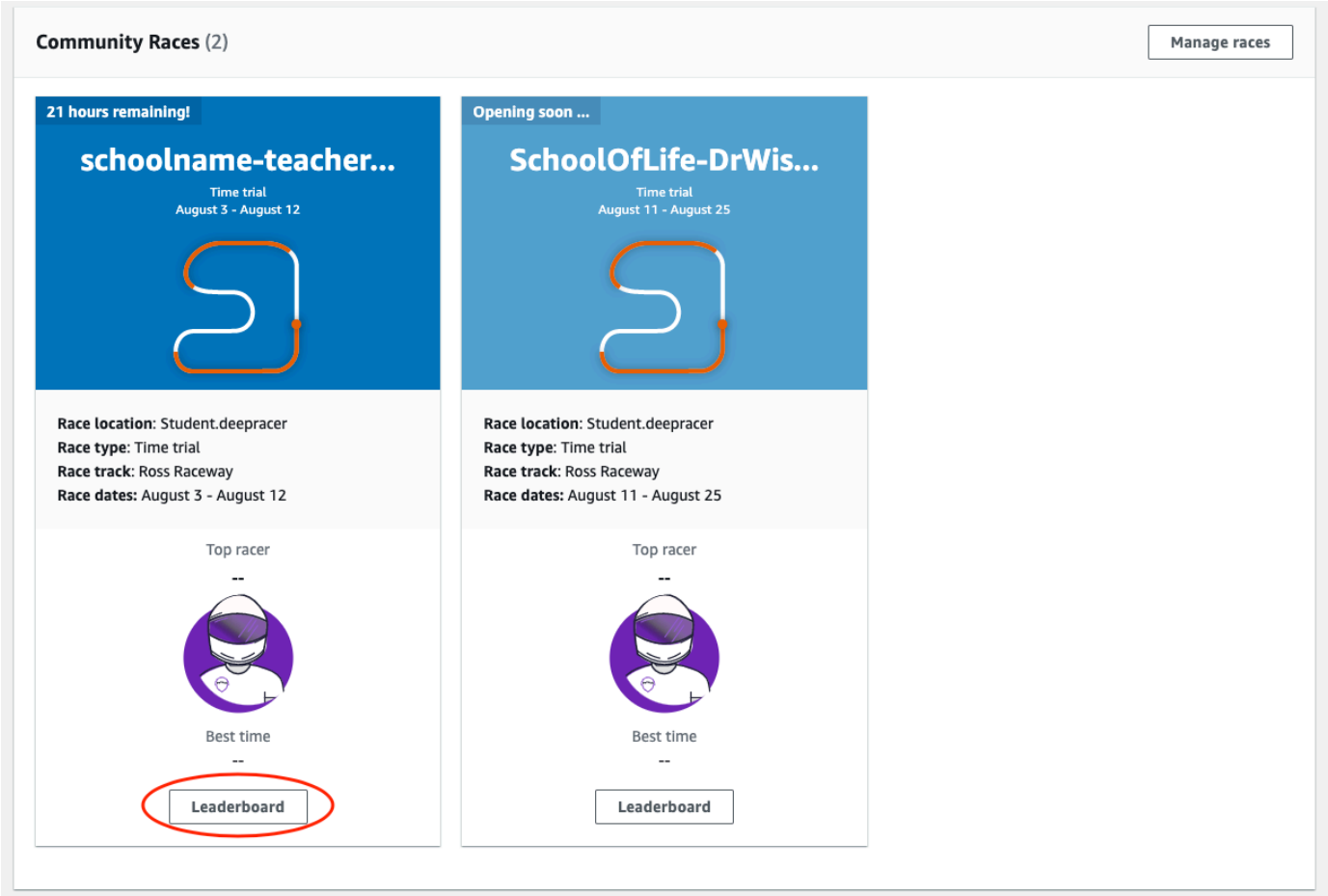
> **Note**
> 레이서 탭과 리더보드에는 학생의 별칭만 표시되므로 어떤 별칭이 어떤 학생과 연결되어 있는지 기록해 두십시오.

AWS DeepRacer Student 커뮤니티 레이스를 사용자 지정합니다.

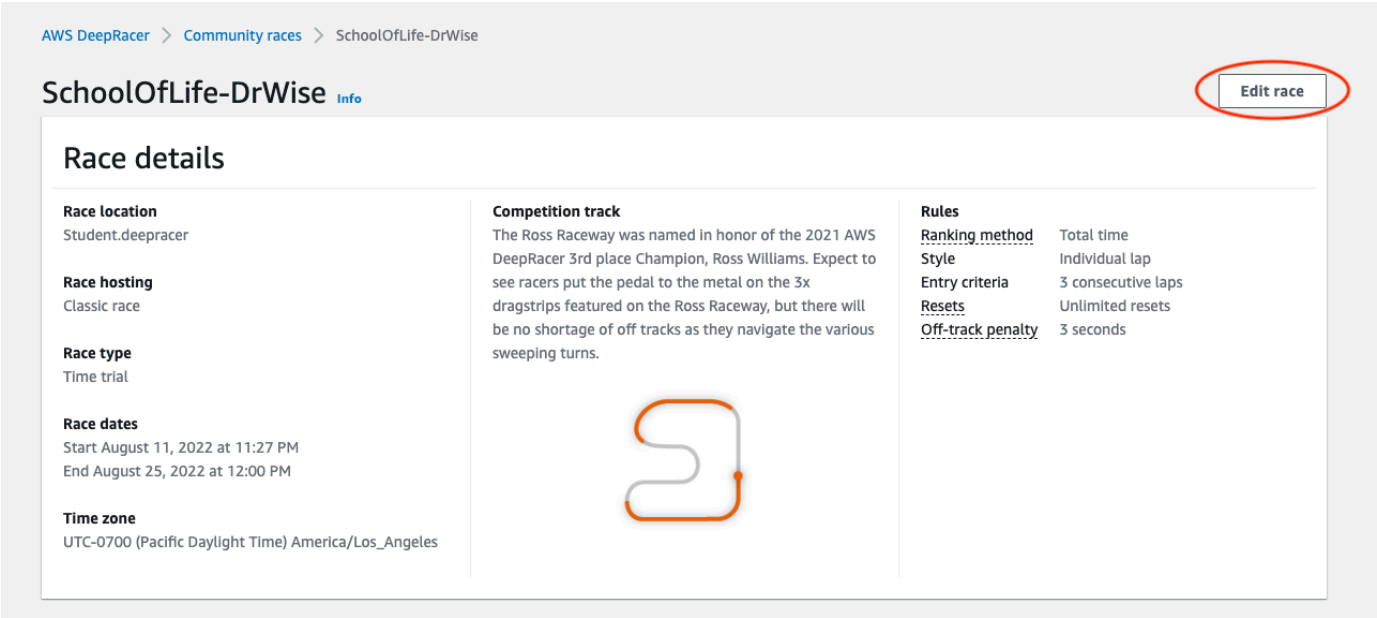
그룹에 맞는 레이스를 만들려면 레이스의 복잡성과 난이도를 높이거나 낮추는 사용자 지정을 추가합니다.

학생 레이스를 사용자 지정하려면

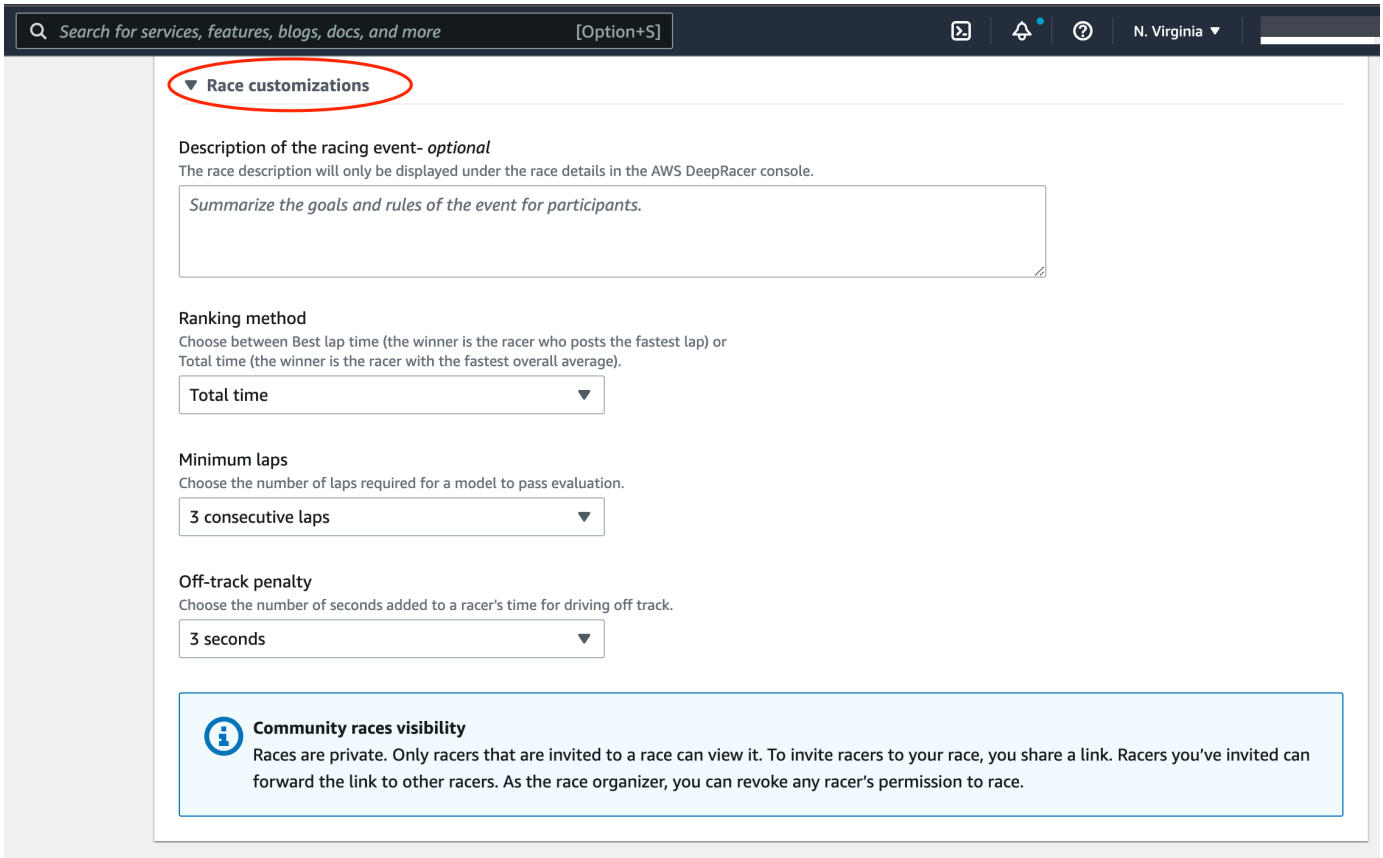
1. AWS DeepRacer 콘솔을 엽니다.
2. 커뮤니티 레이스를 선택합니다.
3. 커뮤니티 레이스 페이지에서 사용자 지정하려는 레이스의 리더보드를 선택하십시오.



4. 레이스 세부 정보 페이지에서 레이스 편집을 선택합니다.



5. 레이스 사용자 지정을 확장하십시오.



6. 참가자를 위해 이벤트의 목표와 규칙을 요약한 레이스에 대한 설명을 작성할 수도 있습니다. 설명은 리더보드 세부 정보에 표시됩니다.
7. 클래식 레이스의 랭킹 방식은 가장 빠른 랩을 기록한 레이서가 우승하는 베스트 랩 타임, 이벤트 기간 내에 여러 번 시도한 결과 평균 시간이 가장 좋은 레이서가 우승하는 평균 시간, 전체 평균 시간이 가장 빠른 레이서가 우승하는 총 시간 중에서 선택합니다.
8. 레이스 리더보드에 결과를 제출할 자격을 얻기 위해 레이서가 완주해야 하는 연속 랩 수인 최소 랩 수를 선택합니다. 초보자 레이스의 경우 작은 숫자를 선택합니다. 고급 사용자의 경우 더 큰 숫자를 선택합니다.
9. 오프 트랙 페널티의 경우 RL 모델이 트랙을 벗어날 때 레이서의 시간에 추가할 시간을 초 단위로 선택합니다.
10. 이제 학생 커뮤니티 레이스를 위한 모든 사용자 지정 옵션을 완료했습니다. 다음을 선택하여 레이스 세부 정보를 검토하십시오.
11. 레이스 세부 정보 검토 페이지에서 레이스 사양을 검토합니다. 변경하려면 편집 또는 이전을 선택하여 레이스 세부 정보 페이지로 돌아가십시오. 초대 링크를 받을 준비가 되면 제출을 선택합니다.
12. 완료 를 선택합니다. 레이스 관리 페이지가 표시됩니다.

이메일 템플릿을 사용하여 새 레이서를 초대하고, 레이스에서 레이서를 삭제하고, 레이스의 모델 제출 상태 등을 확인하는 방법을 알아보려면 커뮤니티 레이스 관리를 참조합니다.

AWS DeepRacer Student 커뮤니티 레이스 관리

모든 학생 커뮤니티 레이스는 초대 링크를 받은 개인만 볼 수 있습니다. 참가자는 초대 링크를 자유롭게 전달할 수 있습니다. 하지만 참가자가 레이스에 참가하려면 AWS DeepRacer Student 계정이 필요합니다. 처음 참가하는 사용자는 계정 생성 프로세스를 완료해야 레이스에 참가할 수 있습니다. 학생은 이메일 주소만 있으면 계정을 만들 수 있습니다.

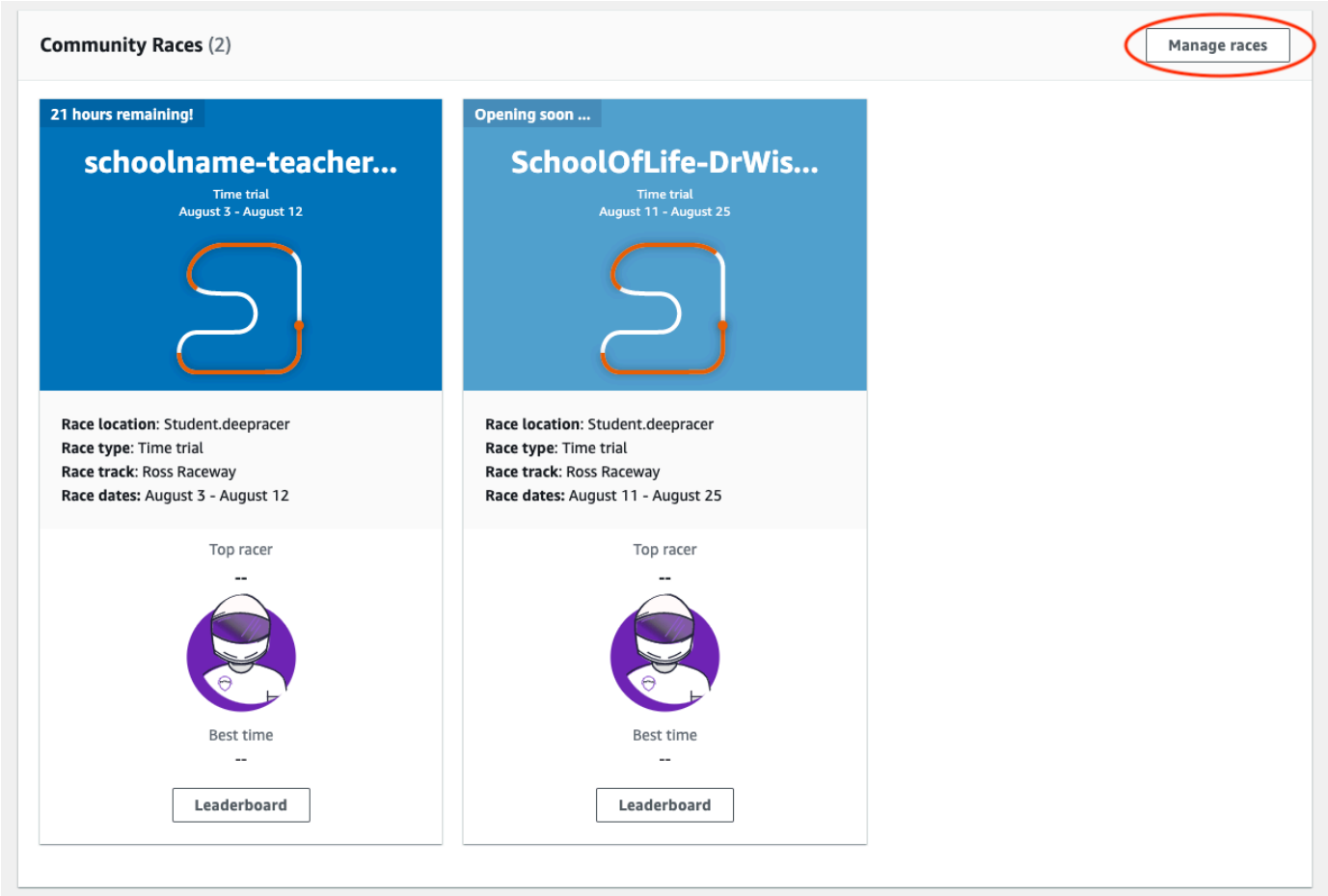
레이스 주최자는 다음을 수행할 수 있습니다.

- 레이스 세부 정보 편집(시작 및 종료 날짜 포함)
- 참가자 제거
- 레이스 종료
- 레이스 삭제

> **Note**
>
> 레이서 탭과 리더보드에는 학생의 별칭만 표시되므로 어떤 별칭이 어떤 학생과 연결되어 있는지 기록해 두십시오.

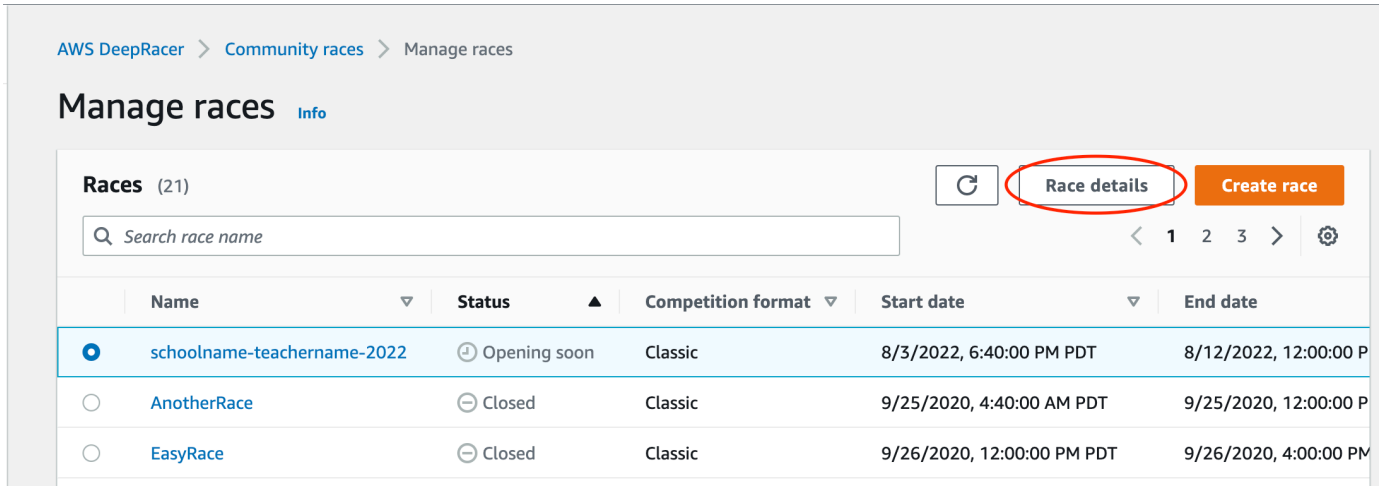
AWS DeepRacer Student 커뮤니티 레이스를 관리하려면

1. AWS DeepRacer 콘솔에 로그인합니다.
2. 커뮤니티 레이스를 선택합니다.
3. 레이스 관리를 선택합니다.



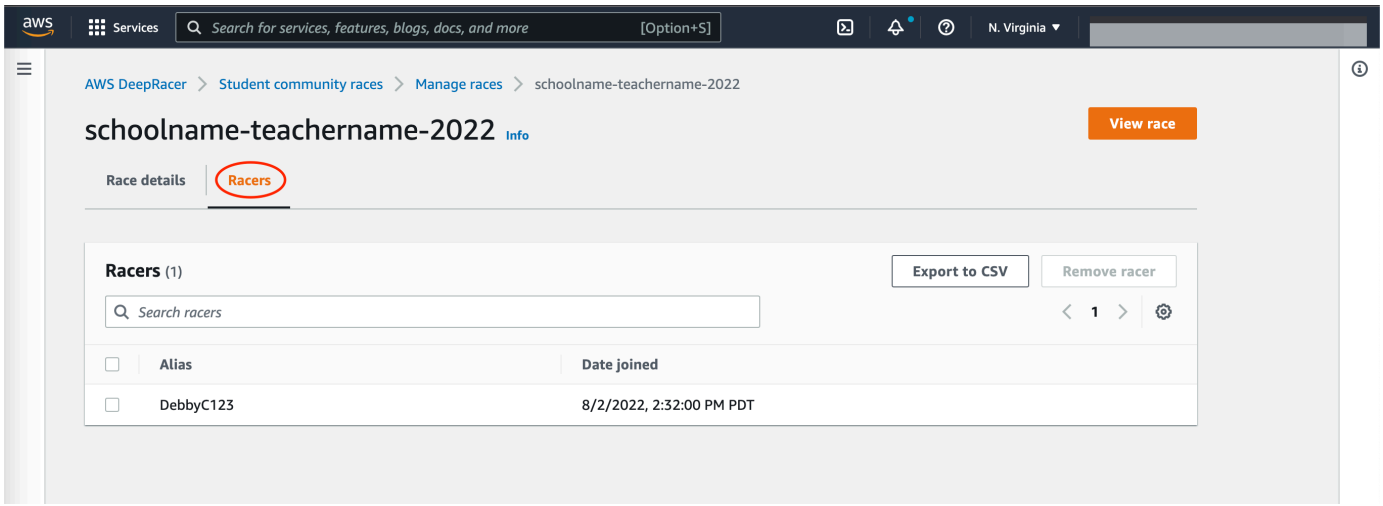
4. 레이스 관리 페이지에서 관리하려는 레이스를 선택합니다.

5. 레이스 세부 정보를 선택하고 편집을 선택합니다.



6. 이벤트의 리더보드를 보려면 레이스 보기를 선택합니다.

7. 이벤트 초대 링크를 재설정하려면 초대 링크 재설정을 선택합니다. 초대 링크를 재설정하면 레이스에서 원본 링크를 아직 선택하지 않은 사람은 입장할 수 없습니다. 초대 링크를 재설정해도 레이스에 참가한 기존 참가자는 영향을 받지 않습니다.
8. 레이스를 종료하려면 레이스 종료를 선택합니다. 그러면 레이스가 즉시 종료됩니다.
9. 이벤트를 삭제하려면 레이스 삭제를 선택합니다. 이렇게 하면 AWS 콘솔 및 AWS DeepRacer Student에서이 레이스가 영구적으로 제거됩니다.
10. 참가자를 제거하려면 레이서 탭을 선택하고 참가자를 한 명 이상 선택한 다음 레이서 제거를 선택합니다. 이벤트에서 참가자를 제거하면 참가자는 레이스에 참가할 수 없습니다.



AWS DeepRacer에 사용되는 보안

AWS DeepRacer를 사용하여 강화 학습을 훈련하고 평가하려면 AWS 계정에 훈련 작업을 실행하는 Amazon VPC와 훈련된 모델 아티팩트를 저장하는 Amazon S3 버킷을 포함하여 종속 AWS 리소스에 액세스할 수 있는 적절한 보안 권한이 있어야 합니다.

AWS DeepRacer 콘솔은 종속 서비스에 필요한 보안 설정을 구성할 수 있는 방법을 제공합니다. 이 섹션에서는 AWS DeepRacer가 의존하는 AWS 서비스와 종속 서비스에 액세스하는 데 필요한 권한을 정의하는 IAM 역할 및 정책을 설명합니다.

주제

- AWS DeepRacer의 데이터 보호
- AWS DeepRacer 종속 AWS 서비스
- AWS DeepRacer가 종속 AWS 서비스를 호출하는 데 필요한 IAM 역할
- AWS Identity and Access Management for AWS DeepRacer

AWS DeepRacer의 데이터 보호

AWS DeepRacer는공 AWS 동 책임 모델 여기에는 데이터 보호에 대한 규정 및 지침이 포함됩니다. AWS 는 모든 AWS 서비스를 실행하는 글로벌 인프라를 보호할 책임이 있습니다. AWS 는이 인프라에서 호스팅되는 데이터에 대한 제어를 유지합니다. 고객 콘텐츠 및 개인 데이터 처리를 위한 보안 구성 제어 포함. AWS 고객 및 APN 파트너, 데이터 컨트롤러 또는 데이터 프로세서로 작동 는 AWS 클라우드에 저장한 모든 개인 데이터에 대해 책임을 집니다.

데이터 보호를 위해 AWS 계정 자격 증명을 보호하고 AWS Identity and Access Management (IAM)를 사용하여 개별 사용자 계정을 설정하여 각 사용자에게 직무를 수행하는 데 필요한 권한만 부여하는 것이 좋습니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용하세요.
- SSL/TLS를 사용하여 AWS 리소스와 통신합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다 AWS CloudTrail.
- AWS 암호화 솔루션과 서비스 내 AWS 모든 기본 보안 제어를 사용합니다.
- Amazon S3에 저장된 개인 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용합니다.

이름 필드와 같은 자유 형식 필드에 고객 계정 번호와 같은 중요 식별 정보를 절대 입력하지 마세요. 여기에는 콘솔 AWS CLI, API 또는 AWS SDKs를 사용하여 AWS DeepRacer 또는 기타 AWS 서비스를 사용하는 경우가 포함됩니다. AWS DeepRacer 또는 기타 서비스에 입력하는 모든 데이터는 진단 로그에 포함하기 위해 선택될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 자격 증명 정보를 URL에 포함시키지 마십시오.

데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하십시오.

AWS DeepRacer 종속 AWS 서비스

AWS DeepRacer에서는 다음 AWS 제품을 사용하여 필요한 리소스를 관리합니다.

Amazon Simple Storage Service(S3)

훈련된 모델 아티팩트를 Amazon S3 버킷에 저장합니다.

AWS Lambda

보상 함수를 생성하고 실행합니다.

AWS CloudFormation

AWS DeepRacer 모델을 위한 훈련 작업을 생성합니다.

SageMaker AI

AWS DeepRacer 모델을 학습시킵니다.

종속 AWS Lambda AWS CloudFormation및 SageMaker AI는 Amazon CloudWatch 및 Amazon CloudWatch Logs를 비롯한 다른 AWS 서비스를 사용합니다.

다음 표에는 AWS DeepRacer에서 직접 또는 간접적으로 사용하는 AWS 서비스가 나와 있습니다.

AWS AWS DeepRacer가 직접 또는 간접적으로 사용하는 서비스

AWS 서비스 보안 주체	설명
application-autoscaling	• SageMaker AI에서 간접적으로 호출하여 작업을 자동으로 확장합니다.
cloudformation	• AWS DeepRacer에서 직접적으로 호출하여 계정 리소스를 생성합니다.

AWS 서비스 보안 주체	설명
cloudwatch	• 작업을 로깅하기 위해 AWS DeepRacer에서 직접적으로 호출합니다. • SageMaker AI가 작업을 로깅하기 위해 간접적으로 호출합니다.
ec2	• AWS CloudFormation 및 SageMaker AI에서 간접적으로 호출하여 훈련 작업을 생성하고 실행합니다.
kinesisvideo	• 캐시된 훈련 스트림을 보기 위해 AWS DeepRacer에서 직접적으로 호출합니다.
lambda	• 보상 함수를 생성 및 실행하기 위해 AWS DeepRacer에서 직접적으로 호출합니다.
logs	• 작업을 로깅하기 위해 AWS DeepRacer에서 직접적으로 호출합니다. • 에서 작업을 로깅 AWS Lambda 하기 위해 간접적으로 호출합니다.
s3	• SageMaker AI에서 간접적으로 호출하여 SageMaker AI별 스토리지 작업을 수행합니다. • AWS DeepRacer에서 직접 호출하여 이름이 ‘deepracer ‘로 시작하는 버킷을 생성, 나열 및 삭제합니다. 또한 버킷에서 객체를 다운로드하거나, 버킷에 객체를 업로드하거나, 버킷에서 객체를 삭제할 때도 직접적으로 호출됩니다.
sagemaker	• 강화 학습 모델을 훈련하기 위해 AWS DeepRacer에서 직접적으로 호출합니다.

AWS DeepRacer를 사용하여 이러한 서비스를 직접적으로 호출하려면 필수 정책이 연결되어 있는 적절한 IAM 역할이 있어야 합니다. AWS DeepRacer가 종속 AWS 서비스를 호출하는 데 필요한 IAM 역할 단원에서는 이러한 정책과 역할에 대해 자세히 알아봅니다.

AWS DeepRacer가 종속 AWS 서비스를 호출하는 데 필요한 IAM 역할

모델을 생성하기 전에 AWS DeepRacer 콘솔을 사용하여 계정에 대한 리소스를 설정합니다. 이렇게 하면 AWS DeepRacer 콘솔에서 다음과 같은 IAM 역할이 생성됩니다.

AWSDeepRacerServiceRole

AWS DeepRacer가 사용자를 대신하여 필요한 리소스를 생성하고 AWS 서비스를 호출하도록 허용합니다.

AWSDeepRacerSageMakerAccessRole

Amazon SageMaker AI가 사용자를 대신하여 필요한 리소스를 생성하고 AWS 서비스를 호출하도록 허용합니다.

AWSDeepRacerLambdaAccessRole

AWS Lambda 함수가 사용자를 대신하여 AWS 서비스를 호출하도록 허용합니다.

AWSDeepRacerCloudFormationAccessRole

AWS CloudFormation 가 사용자를 대신하여 AWS 스택과 리소스를 생성하고 관리할 수 있도록 허용합니다.

링크를 따라 AWS IAM 콘솔에서 자세한 액세스 권한을 봅니다.

AWS Identity and Access Management for AWS DeepRacer

AWS Identity and Access Management (IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어할 수 AWS 서비스 있도록 도와주는 입니다. IAM 관리자는 누가 DeepRacer 리소스를 사용하도록 인증되고(로그인됨) 권한이 부여되는지(권한 있음)를 제어합니다. IAM은 추가 비용 없이 사용할 수 AWS 서비스 있는 입니다.

주제

- 대상
- ID를 통한 인증
- 정책을 사용하여 액세스 관리
- AWS DeepRacer가 IAM과 작동하는 방법
- AWS DeepRacer에 대한 자격 증명 기반 정책 예제
- AWS AWS DeepRacer에 대한 관리형 정책
- 교차 서비스 혼동된 대리인 방지
- 문제 해결 AWS DeepRacer 자격 증명 및 액세스

대상

사용 방법 AWS Identity and Access Management (IAM)은 DeepRacer에서 수행하는 작업에 따라 다릅니다.

서비스 사용자 - DeepRacer 서비스를 사용하여 작업을 수행하는 경우 필요한 자격 증명과 권한을 관리자가 제공합니다. 더 많은 DeepRacer 기능을 사용하여 작업을 수행하게 되면 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방법을 이해하면 관리자에게 올바른 권한을 요청하는 데 도움이 됩니다. DeepRacer의 기능에 액세스할 수 없는 경우 문제 해결 AWS DeepRacer 자격 증명 및 액세스 단원을 참조하십시오.

서비스 관리자 - 회사에서 DeepRacer 리소스를 책임지고 있는 경우 DeepRacer에 대한 전체 액세스 권한을 가지고 있을 것입니다. 서비스 관리자는 서비스 사용자가 액세스해야 하는 DeepRacer 기능과 리소스를 결정합니다. 그런 다음 IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해하십시오. 회사가 DeepRacer에서 IAM을 사용하는 방법에 대해 자세히 알아보려면 AWS DeepRacer가 IAM과 작동하는 방법을(를) 참조하십시오.

IAM 관리자 - IAM 관리자라면 DeepRacer에 대한 액세스 권한 관리 정책 작성 방법을 자세히 알고 싶을 것입니다. IAM에서 사용할 수 있는 DeepRacer 자격 증명 기반 정책 예제를 보려면 AWS DeepRacer에 대한 자격 증명 기반 정책 예제 단원을 참조하십시오.

ID를 통한 인증

인증은 AWS 자격 증명으로에 로그인하는 방법입니다. IAM 사용자 또는 AWS 계정 루트 사용자 IAM 역할을 수임하여 로 인증(로그인 AWS)되어야 합니다.

자격 증명 소스를 통해 제공된 자격 증명을 사용하여 페더레이션 자격 증명 AWS 으로에 로그인할 수 있습니다. AWS IAM Identity Center (IAM Identity Center) 사용자, 회사의 Single Sign-On 인증 및 Google 또는 Facebook 자격 증명은 페더레이션 자격 증명의 예입니다. 페더레이션형 ID로 로그인할 때 관리자가 이전에 IAM 역할을 사용하여 ID 페더레이션을 설정했습니다. 페더레이션을 사용하여 AWS 에 액세스하면 간접적으로 역할을 수임하게 됩니다.

사용자 유형에 따라 AWS Management Console 또는 AWS 액세스 포털에 로그인할 수 있습니다. 에 로그인하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의에 로그인하는 방법을 AWS참조하세요. AWS 계정

AWS 프로그래밍 방식으로에 액세스하는 경우는 자격 증명을 사용하여 요청에 암호화 방식으로 서명할 수 있는 소프트웨어 개발 키트(SDK)와 명령줄 인터페이스(CLI)를 AWS 제공합니다. AWS 도구를 사용하지 않는 경우 직접 요청에 서명해야 합니다. 권장 방법을 사용하여 요청에 직접 서명하는 자세한 방법은 IAM 사용 설명서에서 API 요청용AWS Signature Version 4를 참조하세요.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어는 다중 인증(MFA)을 사용하여 계정의 보안을 강화할 것을 AWS 권장합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 다중 인증 및 IAM 사용 설명서에서 IAM의AWS 다중 인증을 참조하세요.

AWS 계정 루트 사용자

를 생성할 때 계정의 모든 AWS 서비스 및 리소스에 대한 전체 액세스 권한이 있는 하나의 로그인 자격 증명으로 AWS 계정시작합니다. 이 자격 증명을 테 AWS 계정 루트 사용자라고 하며 계정을 생성하는 데 사용한 이메일 주소와 암호로 로그인하여 액세스합니다. 일상적인 작업에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 자격 증명을 보호하고 루트 사용자만 수행할 수 있는 작업을 수행하는 데 사용합니다. 루트 사용자로 로그인해야 하는 전체 작업 목록은 IAM 사용 설명서의 루트 사용자 자격 증명이 필요한 작업을 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 한 사람 또는 애플리케이션에 대한 특정 권한이 AWS 계정 있는 내 자격 증명입니다. 가능하면 암호 및 액세스 키와 같은 장기 자격 증명이 있는 IAM 사용자를 생성하는 대신 임시 자격 증명을 사용하는 것이 좋습니다. 하지만 IAM 사용자의 장기 자격 증명이 필요한 특정 사용 사례가 있는 경우, 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 장기 보안 인증이 필요한 사용 사례의 경우, 정기적으로 액세스 키 교체를 참조하세요.

IAM 그룹은 IAM 사용자 컬렉션을 지정하는 자격 증명입니다. 사용자는 그룹으로 로그인할 수 없습니다. 그룹을 사용하여 여러 사용자의 권한을 한 번에 지정할 수 있습니다. 그룹을 사용하면 대규모 사용자 집합의 권한을 더 쉽게 관리할 수 있습니다. 예를 들어, IAMAdmins라는 그룹이 있고 이 그룹에 IAM 리소스를 관리할 권한을 부여할 수 있습니다.

사용자는 역할과 다릅니다. 사용자는 한 사람 또는 애플리케이션과 고유하게 연결되지만, 역할은 해당 역할이 필요한 사람이라면 누구나 수임할 수 있습니다. 사용자는 영구적인 장기 자격 증명을 가지고 있지만, 역할은 임시 보안 인증만 제공합니다. 자세한 내용은 IAM 사용 설명서에서 IAM 사용자 사용 사례를 참조하세요.

IAM 역할

IAM 역할은 특정 권한이 AWS 계정 있는 내 자격 증명입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. 에서 IAM 역할을 일시적으로 수임하려면 사용자에서 IAM 역할(콘솔)로 전환할 AWS Management Console수 있습니다. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-console.html 또는 AWS API 작업을 호출하거나 사용자 지정 URL을 AWS CLI 사용하여 역할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 내용은 IAM 사용 설명서의 역할 수임 방법을 참조하세요.

임시 보안 인증이 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

- 페더레이션 사용자 액세스 - 페더레이션 ID에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 페더레이션 ID가 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션 관련 역할에 대한 자세한 내용은 IAM 사용 설명서의 Create a role for a third-party identity provider (federation)를 참조하세요. IAM Identity Center를 사용하는 경우, 권한 집합을 구성합니다. 인증 후 ID가 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 집합을 IAM의 역할과 연관짓습니다. 권한 집합에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 권한 집합을 참조하세요.
- 임시 IAM 사용자 권한 - IAM 사용자 또는 역할은 IAM 역할을 수임하여 특정 작업에 대한 다양한 권한을 임시로 받을 수 있습니다.
- 교차 계정 액세스 - IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입니다. 그러나 일부 에서는 (역할을 프록시로 사용하는 대신) 정책을 리소스에 직접 연결할 AWS 서비스 수 있습니다. 교차 계정 액세스에 대한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 교차 계정 리소스 액세스를 참조하세요.
- 교차 서비스 액세스 - 일부는 다른에서 기능을 AWS 서비스 사용합니다 AWS 서비스. 예를 들어, 서비스에서 호출하면 일반적으로 해당 서비스는 Amazon EC2에서 애플리케이션을 실행하거나 Amazon S3에 객체를 저장합니다. 서비스는 직접적으로 호출하는 위탁자의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 작업을 수행할 수 있습니다.
  - 전달 액세스 세션(FAS) - IAM 사용자 또는 역할을 사용하여에서 작업을 수행하는 경우 AWS보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우, 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는를 호출하는 보안 주체의 권한을 다운스트림 서비스에 AWS 서비스 대

한 요청과 AWS 서비스함께 사용합니다. FAS 요청은 서비스가 다른 AWS 서비스 또는 리소스와의 상호 작용을 완료해야 하는 요청을 수신할 때만 수행됩니다. 이 경우, 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.

- 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하기 위해 맡는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 Create a role to delegate permissions to an AWS 서비스를 참조하세요.
- 서비스 연결 역할 - 서비스 연결 역할은에 연결된 서비스 역할의 한 유형입니다 AWS 서비스. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은에 표시 AWS 계정 되며 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

- Amazon EC2에서 실행되는 애플리케이션 - IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 AWS CLI 또는 AWS API 요청을 수행하는 애플리케이션의 임시 자격 증명을 관리할 수 있습니다. 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS 역할을 할당하고 모든 애플리케이션에서 사용할 수 있도록 하려면 인스턴스에 연결된 인스턴스 프로파일을 생성합니다. 인스턴스 프로필에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 보안 인증을 얻을 수 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하세요.

정책을 사용하여 액세스 관리

정책을 AWS 생성하고 자격 증명 또는 리소스에 연결하여 AWS 에서 액세스를 제어합니다. 정책은 자격 증명 또는 리소스와 연결된 AWS 경우 권한을 정의하는의 객체입니다.는 보안 주체(사용자, 루트 사용자 또는 역할 세션)가 요청할 때 이러한 정책을 AWS 평가합니다. 정책에서 권한은 요청이 허용되거나 거부되는 지를 결정합니다. 대부분의 정책은에 JSON 문서 AWS 로 저장됩니다. JSON 정책 문서의 구조와 콘텐츠에 대한 자세한 정보는 IAM 사용 설명서의 JSON 정책 개요를 참조하세요.

관리자는 AWS JSON 정책을 사용하여 누가 무엇을 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

기본적으로, 사용자 및 역할에는 어떠한 권한도 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 수임할 수 있습니다.

IAM 정책은 작업을 수행하기 위해 사용하는 방법과 상관없이 작업에 대한 권한을 정의합니다. 예를 들어, `iam:GetRole` 작업을 허용하는 정책이 있다고 가정합니다. 해당 정책이 있는 사용자는 AWS Management Console AWS CLI, 또는 API에서 역할 정보를 가져올 수 있습니다 AWS .

자격 증명 기반 정책

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자 및 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

ID 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 사용자, 그룹 또는 역할에 직접 포함됩니다. 관리형 정책은의 여러 사용자, 그룹 및 역할에 연결할 수 있는 독립 실행형 정책입니다 AWS 계정. 관리형 정책에는 AWS 관리형 정책 및 고객 관리형 정책이 포함됩니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책 및 인라인 정책 중에서 선택을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예제는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 위탁자가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 위탁자를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는가 포함될 수 있습니다 AWS 서비스.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책에서는 IAM의 AWS 관리형 정책을 사용할 수 없습니다.

액세스 제어 목록(ACL)

액세스 제어 목록(ACL)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACL은 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

Amazon S3 AWS WAF및 Amazon VPC는 ACLs. ACL에 관한 자세한 내용은 Amazon Simple Storage Service 개발자 가이드의 액세스 제어 목록(ACL) 개요를 참조하세요.

기타 정책 유형

AWS 는 덜 일반적인 추가 정책 유형을 지원합니다. 이러한 정책 타입은 더 일반적인 정책 유형에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 – 권한 경계는 ID 기반 정책에 따라 IAM 엔터티(IAM 사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 개체에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻

는 권한은 객체의 ID 기반 정책과 그 권한 경계의 교집합입니다. `Principal` 필드에서 사용자나 역할을 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 정보는 IAM 사용 설명서의 IAM 엔티티에 대한 권한 경계를 참조하세요.

- 서비스 제어 정책(SCPs) - SCPs는의 조직 또는 조직 단위(OU)에 대한 최대 권한을 지정하는 JSON 정책입니다 AWS Organizations. AWS Organizations 는 비즈니스가 소유 AWS 계정 한 여러를 그룹화하고 중앙에서 관리하기 위한 서비스입니다. 조직에서 모든 기능을 활성화할 경우, 서비스 제어 정책(SCP)을 임의의 또는 모든 계정에 적용할 수 있습니다. SCP는 각각을 포함하여 멤버 계정의 엔터티에 대한 권한을 제한합니다 AWS 계정 루트 사용자. 조직 및 SCP에 대한 자세한 내용은 AWS Organizations 사용 설명서에서 Service control policies을 참조하세요.
- 리소스 제어 정책(RCP) - RCP는 소유한 각 리소스에 연결된 IAM 정책을 업데이트하지 않고 계정의 리소스에 대해 사용 가능한 최대 권한을 설정하는 데 사용할 수 있는 JSON 정책입니다. RCP는 멤버 계정의 리소스에 대한 권한을 제한하며 조직에 속하는지 여부에 AWS 계정 루트 사용자관계없이를 포함한 자격 증명에 대한 유효 권한에 영향을 미칠 수 있습니다. RCP를 AWS 서비스 지원하는 목록을 포함하여 조직 및 RCPs에 대한 자세한 내용은 AWS Organizations 사용 설명서의 리소스 제어 정책(RCPs)을 참조하세요.
- 세션 정책 – 세션 정책은 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할의 ID 기반 정책의 교차와 세션 정책입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 정보는 IAM 사용 설명서의 세션 정책을 참조하세요.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 가 여러 정책 유형이 관련될 때 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하세요.

AWS DeepRacer가 IAM과 작동하는 방법

IAM을 사용하여 DeepRacer에 대한 액세스를 관리하기 전에 DeepRacer와 함께 사용할 수 있는 IAM 기능을 알아보십시오.

AWS DeepRacer와 함께 사용할 수 있는 IAM 기능

IAM 기능	DeepRacer 지원
ID 기반 정책	예
리소스 기반 정책	아니요
정책 작업	예
정책 리소스	예
정책 조건 키	예
ACLs	아니요
ABAC(정책의 태그)	예
임시 보안 인증	예
보안 주체 권한	예
서비스 역할	예
서비스 연결 역할	아니요

DeepRacer 및 기타 AWS 서비스가 대부분의 IAM 기능과 작동하는 방식을 전체적으로 알아보려면 IAM 사용 설명서의 AWS IAM으로 작업하는 서비스를 참조하세요.

DeepRacer에 대한 자격 증명 기반 정책

ID 기반 정책 지원: 예

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자 및 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

IAM ID 기반 정책을 사용하면 허용되거나 거부되는 작업과 리소스뿐 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. ID 기반 정책에서는 위탁자가 연결된 사용자 또는 역할에 적용되므로

위탁자를 지정할 수 없습니다. JSON 정책에서 사용하는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하십시오.

DeepRacer 자격 증명 기반 정책 예제

DeepRacer 자격 증명 기반 정책의 예를 보려면 AWS DeepRacer에 대한 자격 증명 기반 정책 예제 단원을 참조하십시오.

DeepRacer 내 리소스 기반 정책

리소스 기반 정책 지원: 아니요

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예제는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 위탁자가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 위탁자를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는가 포함될 수 있습니다 AWS 서비스.

교차 계정 액세스를 활성화하려는 경우, 전체 계정이나 다른 계정의 IAM 개체를 리소스 기반 정책의 위탁자로 지정할 수 있습니다. 리소스 기반 정책에 크로스 계정 보안 주체를 추가하는 것은 트러스트 관계 설정의 절반밖에 되지 않는다는 것을 유념하세요. 보안 주체와 리소스가 다른 경우 신뢰할 수 AWS 계정있는 계정의 IAM 관리자는 보안 주체 엔터티(사용자 또는 역할)에게 리소스에 액세스할 수 있는 권한도 부여해야 합니다. 엔터티에 ID 기반 정책을 연결하여 권한을 부여합니다. 하지만 리소스 기반 정책이 동일 계정의 위탁자에 액세스를 부여하는 경우, 추가 자격 증명 기반 정책이 필요하지 않습니다. 자세한 내용은 IAM 사용 설명서의 교차 계정 리소스 액세스를 참조하세요.

DeepRacer의 정책 작업

정책 작업 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇을 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 위탁자가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 `Action` 요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 작업을 설명합니다. 정책 작업은 일반적으로 연결된 AWS API 작업과 이름이 동일합니다. 일치하는 API 작업이 없는 권한 전용 작업 같은 몇 가지 예외도 있습니다. 정책에서 여러 작업이 필요한 몇 가지 작업도 있습니다. 이러한 추가 작업을 일컬어 종속 작업이라고 합니다.

연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하세요.

DeepRacer 작업 목록을 보려면 서비스 승인 참조의AWS DeepRacer에서 정의한 작업을 참조하세요.

DeepRacer의 정책 작업은 작업 앞에 다음 접두사를 사용합니다.

```
deepracer
```

단일 문에서 여러 작업을 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
      "deepracer:action1",
      "deepracer:action2"
         ]
```

DeepRacer 자격 증명 기반 정책의 예를 보려면 AWS DeepRacer에 대한 자격 증명 기반 정책 예제 단원을 참조하십시오.

DeepRacer 정책 리소스

정책 리소스 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇을 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Resource` JSON 정책 요소는 작업이 적용되는 하나 이상의 객체를 지정합니다. 문에는 `Resource`또는 `NotResource`요소가 반드시 추가되어야 합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한이라고 하는 특정 리소스 유형을 지원하는 작업에 대해 이를 수행할 수 있습니다.

작업 나열과 같이 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

DeepRacer 리소스 유형 및 해당 ARNs의 목록을 보려면 서비스 승인 참조의AWS DeepRacer에서 정의한 리소스를 참조하세요. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면AWS DeepRacer에서 정의한 작업을 참조하세요.

DeepRacer 자격 증명 기반 정책의 예를 보려면 AWS DeepRacer에 대한 자격 증명 기반 정책 예제 단원을 참조하십시오.

DeepRacer에서 사용되는 정책 조건 키

서비스별 정책 조건 키 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇을 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Condition` 요소(또는 `Condition` 블록)를 사용하면 정책이 발효되는 조건을 지정할 수 있습니다. `Condition` 요소는 옵션입니다. 같거나 작음과 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다.

한 문에서 여러 `Condition` 요소를 지정하거나 단일 `Condition` 요소에서 여러 키를 지정하는 경우, AWS 는 논리적 `AND` 작업을 사용하여 평가합니다. 단일 조건 키에 여러 값을 지정하는 경우는 논리적 `OR` 작업을 사용하여 조건을 AWS 평가합니다. 문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리 표시자 변수를 사용할 수도 있습니다. 예를 들어, IAM 사용자에게 IAM 사용자 이름으로 태그가 지정된 경우에만 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM 정책 요소: 변수 및 태그를 참조하세요.

AWS 는 전역 조건 키와 서비스별 조건 키를 지원합니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

DeepRacer 조건 키 목록을 보려면 서비스 권한 부여 참조의 IAM 사용 설명서에서AWS DeepRacer에 사용되는 조건 키를 참조하세요. 조건 키를 사용할 수 있는 작업 및 리소스를 알아보려면AWS DeepRacer에서 정의한 작업을 참조하세요.

DeepRacer 자격 증명 기반 정책의 예를 보려면 AWS DeepRacer에 대한 자격 증명 기반 정책 예제 단원을 참조하십시오.

DeepRacer의 액세스 제어 목록(ACL)

ACL 지원: 아니요

액세스 제어 목록(ACL)은 어떤 위탁자(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACL은 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

DeepRacer의 ABAC(속성 기반 액세스 제어)

ABAC 지원(정책의 태그): 예

속성 기반 액세스 제어(ABAC)는 속성에 근거하여 권한을 정의하는 권한 부여 전략입니다. 에서는 AWS이러한 속성을 태그라고 합니다. IAM 엔터티(사용자 또는 역할) 및 많은 AWS 리소스에 태그를 연결할 수 있습니다. ABAC의 첫 번째 단계로 개체 및 리소스에 태그를 지정합니다. 그런 다음 위탁자의 태그가 액세스하려는 리소스의 태그와 일치할 때 작업을 허용하도록 ABAC 정책을 설계합니다.

ABAC는 빠르게 성장하는 환경에서 유용하며 정책 관리가 번거로운 상황에 도움이 됩니다.

태그에 근거하여 액세스를 제어하려면 `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` 또는 `aws:TagKeys` 조건 키를 사용하여 정책의 조건 요소에 태그 정보를 제공합니다.

서비스가 모든 리소스 유형에 대해 세 가지 조건 키를 모두 지원하는 경우, 값은 서비스에 대해 예입니다. 서비스가 일부 리소스 유형에 대해서만 세 가지 조건 키를 모두 지원하는 경우, 값은 부분적입니다.

ABAC에 대한 자세한 내용은 IAM 사용 설명서의 ABAC 권한 부여를 통한 권한 정의를 참조하세요. ABAC 설정 단계가 포함된 자습서를 보려면 IAM 사용 설명서의 속성 기반 액세스 제어(ABAC) 사용을 참조하세요.

DeepRacer에서 임시 자격 증명 사용

임시 자격 증명 지원: 예

임시 자격 증명을 사용하여 로그인할 때 작동하지 AWS 서비스 않는 경우도 있습니다. 임시 자격 증명으로 AWS 서비스 작업하는을 비롯한 자세한 내용은 AWS 서비스 IAM 사용 설명서의 IAM으로 작업하는를 참조하세요.

사용자 이름과 암호를 제외한 방법을 AWS Management Console 사용하여에 로그인하는 경우 임시 자격 증명을 사용합니다. 예를 들어 회사의 SSO(Single Sign-On) 링크를 AWS 사용하여에 액세스하면 해당 프로세스가 임시 자격 증명을 자동으로 생성합니다. 또한 콘솔에 사용자로 로그인한 다음 역할을 전환할 때 임시 자격 증명을 자동으로 생성합니다. 역할 전환에 대한 자세한 내용은 IAM 사용 설명서의 사용자에서 IAM 역할로 전환(콘솔)을 참조하세요.

AWS CLI 또는 AWS API를 사용하여 임시 자격 증명을 수동으로 생성할 수 있습니다. 그런 다음 이러한 임시 자격 증명을 사용하여 장기 액세스 키를 사용하는 대신 동적으로 임시 자격 증명을 생성하는 access AWS. AWS recommends에 액세스할 수 있습니다. 자세한 정보는 IAM의 임시 보안 자격 증명 섹션을 참조하십시오.

DeepRacer의 서비스 간 보안 주체 권한

전달 액세스 세션(FAS) 지원: 예

IAM 사용자 또는 역할을 사용하여에서 작업을 수행하는 경우 AWS보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우, 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는를 호출하는 보안 주체의 권한을 다운스트림 서비스에 AWS 서비스 대한 요청과 AWS 서비스함께 사용합니다. FAS 요청은 서비스가 다른 AWS 서비스 또는 리소스와의 상호 작용을 완료해야 하는 요청을 수신할 때만 수행됩니다. 이 경우, 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.

DeepRacer의 서비스 역할

서비스 역할 지원: 예

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하는 것으로 가정하는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 Create a role to delegate permissions to an AWS 서비스를 참조하세요.

> **Warning**
> 서비스 역할에 대한 권한을 변경하면 DeepRacer 기능이 중단될 수 있습니다. DeepRacer에서 관련 지침을 제공하는 경우에만 서비스 역할을 편집하십시오.

DeepRacer의 서비스 연결 역할

서비스 링크 역할 지원: 아니요

서비스 연결 역할은에 연결된 서비스 역할의 한 유형입니다 AWS 서비스. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은에 표시 AWS 계정 되며 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

서비스 연결 역할 생성 또는 관리에 대한 자세한 내용은 IAM으로 작업하는AWS 서비스를 참조하세요. 서비스 연결 역할 열에서 `Yes`이(가) 포함된 서비스를 테이블에서 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

AWS DeepRacer에 대한 자격 증명 기반 정책 예제

기본적으로 사용자 및 역할에는 DeepRacer 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 또한 AWS Management Console, AWS Command Line Interface (AWS CLI) 또는 AWS API를 사용하여 작업을 수행할 수 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려

면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 맡을 수 있습니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성(콘솔)을 참조하세요.

각 리소스 유형에 대한 ARNs 형식을 포함하여 DeepRacer에서 정의한 작업 및 리소스 유형에 대한 자세한 내용은 서비스 권한 부여 참조의AWS DeepRacer에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

주제

- 정책 모범 사례
- DeepRacer 콘솔 사용
- 사용자가 자신의 고유한 권한을 볼 수 있도록 허용

정책 모범 사례

ID 기반 정책에 따라 계정에서 사용자가 DeepRacer 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. ID 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따릅니다.

- AWS 관리형 정책을 시작하고 최소 권한으로 전환 - 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반적인 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. 에서 사용할 수 있습니다 AWS 계정. 사용 사례에 맞는 AWS 고객 관리형 정책을 정의하여 권한을 추가로 줄이는 것이 좋습니다. 자세한 정보는 IAM 사용 설명서의 AWS 관리형 정책 또는 AWS 직무에 대한 관리형 정책을 참조하세요.
- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우, 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서에 있는 IAM의 정책 및 권한을 참조하세요.
- IAM 정책의 조건을 사용하여 액세스 추가 제한 – 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어, SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 작성할 수 있습니다. 조건을 사용하여 AWS 서비스와 같은 특정를 통해 사용되는 경우 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다 AWS CloudFormation. 자세한 정보는 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 확인하여 안전하고 기능적인 권한 보장 - IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 새로운 및 기존 정책을

확인합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 내용은 IAM 사용 설명서의 IAM Access Analyzer에서 정책 검증을 참조하세요.

- 다중 인증(MFA) 필요 -에서 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 MFA를 AWS 계정켭니다. API 작업을 직접 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 내용은 IAM 사용 설명서의 MFA를 통한 보안 API 액세스를 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

DeepRacer 콘솔 사용

AWS DeepRacer 콘솔에 액세스하려면 최소 권한 집합이 있어야 합니다. 이러한 권한은 AWS 계정에서 DeepRacer 리소스에 대한 세부 정보를 나열하고 볼 수 있도록 허용해야 합니다. 최소 필수 권한보다 더 제한적인 ID 기반 정책을 생성하는 경우, 콘솔이 해당 정책에 연결된 엔티티(사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

AWS CLI 또는 AWS API에만 호출하는 사용자에 대해 최소 콘솔 권한을 허용할 필요는 없습니다. 대신 수행하려는 API 작업과 일치하는 작업에만 액세스할 수 있도록 합니다.

사용자와 역할이 DeepRacer 콘솔을 계속 사용할 수 있도록 하려면 DeepRacer `ConsoleAccess` 또는 `ReadOnly` AWS 관리형 정책도 엔터티에 연결합니다. 자세한 내용은 IAM 사용 설명서의 사용자에게 권한 추가를 참조하십시오.

사용자가 자신의 고유한 권한을 볼 수 있도록 허용

이 예제는 IAM 사용자가 자신의 사용자 ID에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 AWS CLI 또는 AWS API를 사용하여 프로그래밍 방식으로이 작업을 완료할 수 있는 권한이 포함되어 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
```

```
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS AWS DeepRacer에 대한 관리형 정책

AWS 관리형 정책은에서 생성하고 관리하는 독립 실행형 정책입니다 AWS. AWS 관리형 정책은 사용자, 그룹 및 역할에 권한 할당을 시작할 수 있도록 많은 일반적인 사용 사례에 대한 권한을 제공하도록 설계되었습니다.

AWS 관리형 정책은 모든 AWS 고객이 사용할 수 있으므로 특정 사용 사례에 대해 최소 권한 권한을 부여하지 않을 수 있습니다. 사용 사례에 고유한 고객 관리형 정책을 정의하여 권한을 줄이는 것이 좋습니다.

AWS 관리형 정책에 정의된 권한은 변경할 수 없습니다. 가 AWS 관리형 정책에 정의된 권한을 AWS 업데이트하면 정책이 연결된 모든 보안 주체 자격 증명(사용자, 그룹 및 역할)에 영향을 미칩니다. AWS 는 새 AWS 서비스 가 시작되거나 기존 서비스에 새 API 작업을 사용할 수 있게 되면 AWS 관리형 정책을 업데이트할 가능성이 높습니다.

자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책을 참조하세요.

다음 AWS관리형 정책은 AWS DeepRacer 다중 사용자 모드를 사용하여 AWS 계정에서 여러 참가자를 후원하는 데에만 적용됩니다.

- `AWSDeepRacerAccountAdminAccess`은 다중 사용자 계정 관리자에게 필요한 AWS DeepRacer 권한을 부여합니다.
- `AWSDeepRacerDefaultMultiUserAccess`은 AWS DeepRacer 콘솔을 사용하기 위해 필요한 AWS DeepRacer 권한을 부여합니다.

주제

- AWS DeepRacer 관리자를 위한 AWSDeepRacerAccountAdminAccess 관리형 정책
- AWS DeepRacer 다중 사용자 레이서에 대한 AWSDeepRacerDefaultMultiUserAccess 관리형 정책
- AWS 관리형 정책에 대한 AWS DeepRacer 업데이트

AWS DeepRacer 관리자를 위한 AWSDeepRacerAccountAdminAccess 관리형 정책

AWS DeepRacer에서 AWS 계정 ID 및 결제 정보를 사용하도록 여러 프로필을 활성화하려면 `AWSDeepRacerAccountAdminAccess` 정책을 연결합니다.

다른 레이서를 후원하는 데 사용하려는 IAM ID에 `AWSDeepRacerAccountAdminAccess` 정책을 연결할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DeepRacerAdminAccessStatement",
            "Effect": "Allow",
            "Action": [
                "deepracer:*"
            ],
            "Resource": [
                "*"
            ],
            "Condition":
            {
                "Null": {
                    "deepracer:UserToken": "true"
                }
            }
```

```
        }
    ]
}
```

AWS DeepRacer 다중 사용자 레이서에 대한 AWSDeepRacerDefaultMultiUserAccess 관리형 정책

`AWSDeepRacerDefaultMultiUserAccess` 정책은 AWS DeepRacer 레이서에게 다중 사용자 계정 관리자 작업을 제외한 모든 AWS DeepRacer 작업에 대한 액세스 권한을 부여합니다.

계정을 통해 후원하려는 참가자의 IAM ID에 `AWSDeepRacerDefaultMultiUserAccess` 정책을 첨부할 수 있습니다.

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "deepracer:Add*",
    "deepracer:Remove*",
    "deepracer:Create*",
    "deepracer:Perform*",
    "deepracer:Clone*",
    "deepracer:Get*",
    "deepracer:List*",
    "deepracer:Edit*",
    "deepracer:Start*",
    "deepracer:Set*",
    "deepracer:Update*",
    "deepracer:Delete*",
    "deepracer:Stop*",
    "deepracer:Import*",
    "deepracer:Tag*",
    "deepracer:Untag*"
   ],
   "Resource": [
    "*"
   ],
   "Condition": {
    "Null": {
     "deepracer:UserToken": "false"
    },
```

```
    "Bool": {
     "deepracer:MultiUser": "true"
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": [
    "deepracer:GetAccountConfig",
    "deepracer:GetTrack",
    "deepracer:ListTracks",
    "deepracer:TestRewardFunction"
   ],
   "Resource": [
    "*"
   ]
  },
  {
   "Effect": "Deny",
   "Action": [
    "deepracer:Admin*"
   ],
   "Resource": [
    "*"
   ]
  }
 ]
}
```

AWS 관리형 정책에 대한 AWS DeepRacer 업데이트

이 서비스가 이러한 변경 사항을 추적하기 시작한 이후 AWS DeepRacer의 AWS 관리형 정책 업데이트에 대한 세부 정보를 봅니다. 페이지의 변경 사항에 대한 자동 알림을 받으려면 AWS DeepRacer 문서 기록 페이지에서 RSS 피드를 구독하십시오.

변경 사항	설명	날짜
`AWSDeepRacerAccountAdminAccess` 및	새로운 관리형 정책이 추가되어 다중 사용자 모드를 사용하	2021년 10월 26일

변경 사항	설명	날짜
`AWSDeepRacerDefaultMultiUserAccess` 정책 추가	여 하나의 AWS DeepRacer 계정으로 여러 참가자를 후원할 수 있습니다.	
AWS DeepRacer는 정책 변경 사항을 추적하기 시작했습니다.	AWS DeepRacer는 AWS 관리형 정책에 대한 변경 사항을 추적하기 시작했습니다.	2021년 10월 26일

교차 서비스 혼동된 대리인 방지

혼동된 대리자 문제는 작업을 수행할 권한이 없는 엔터티가 권한이 더 많은 엔터티에게 작업을 수행하도록 강요할 수 있는 보안 문제입니다. 에서 AWS교차 서비스 위장은 혼동된 대리자 문제를 초래할 수 있습니다. 교차 서비스 가장은 한 서비스(호출하는 서비스)가 다른 서비스(호출되는 서비스)를 직접적으로 호출할 때 발생할 수 있습니다. 직접적으로 호출하는 서비스는 다른 고객의 리소스에 대해 액세스 권한이 없는 방식으로 작동하게 권한을 사용하도록 조작될 수 있습니다. 이를 방지하기 위해 AWS 에서는 계정의 리소스에 대한 액세스 권한이 부여된 서비스 보안 주체를 사용하여 모든 서비스에 대한 데이터를 보호하는 데 도움이 되는 도구를 제공합니다.

AWSDeepRacerLong가 리소스에 다른 서비스를 제공하는 권한을 제한하려면 리소스 정책에서 `aws:SourceArn` 및 `aws:SourceAccount` 글로벌 조건 컨텍스트 키를 사용하는 것이 좋습니다. 두 전역 조건 컨텍스트 키를 모두 사용하는 경우 `aws:SourceAccount` 값과 `aws:SourceArn` 값의 계정은 동일한 정책 문에서 사용할 경우 동일한 계정 ID를 사용해야 합니다.

`aws:SourceArn`의 값은 s3:::your-bucket-name이어야 합니다.

혼동된 대리인 문제로부터 보호하는 가장 효과적인 방법은 리소스의 전체 ARN이 포함된 `aws:SourceArn` 글로벌 조건 컨텍스트 키를 사용하는 것입니다. 리소스의 전체 ARN을 모를 경우 또는 여러 리소스를 지정하는 경우, ARN의 알 수 없는 부분에 대해 와일드카드(*)를 포함한 `aws:SourceArn` 전역 조건 컨텍스트 키를 사용합니다. 예: `arn:aws:servicename::123456789012:*`.

다음 예는 AWSDeepRacer에서 `aws:SourceArn` 및 `aws:SourceAccount` 글로벌 조건 컨텍스트 키를 사용하여 혼동된 대리자 문제를 방지하는 방법을 보여줍니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "Stmt1586917903457",
            "Effect": "Allow",
            "Principal": {
                "Service": "deepracer.amazonaws.com"
                },
            "Action": [
                "s3:GetObjectAcl",
                "s3:GetObject",
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": [
               "arn:aws:s3:::your-bucket-name",
               "arn:aws:s3:::your-bucket-name/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:SourceArn": "arn:${Partition}:deepracer:${Region}:
${Account}:model/reinforcement_learning/${ResourceId}"
                }
            }
        }
    ]
}
```

이 버킷에 대해 사용자 지정 AWS Key Management Service (KMS) 리소스를 사용하는 경우 AWS KMS 리소스 정책을 포함합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "statement identifier",
    "Effect": "effect",
    "Principal": "principal",
    "Action": "action",
    "Resource": "resource",
    "Condition": {
        "StringEquals": {
            "aws:SourceArn": "arn:${Partition}:deepracer:${Region}:${Account}:model/
reinforcement_learning/${ResourceId}"
        }
    }
```

```
  }]
}
```

문제 해결 AWS DeepRacer 자격 증명 및 액세스

다음 정보를 사용하여 DeepRacer 및 IAM에서 발생할 수 있는 공통적인 문제를 진단하고 수정할 수 있습니다.

주제

- DeepRacer 다중 사용자 계정 모드에서 인증 오류가 발생합니다
- DeepRacer에서 작업을 수행할 권한이 없음
- iam:PassRole을 수행하도록 인증되지 않음
- 액세스 키를 보아야 합니다.
- 관리자가 다른 사용자의 DeepRacer 액세스를 허용하려고 합니다
- 내 외부의 사람이 내 DeepRacer 리소스 AWS 계정 에 액세스하도록 허용하고 싶습니다.

DeepRacer 다중 사용자 계정 모드에서 인증 오류가 발생합니다

AWSDeepracerAccountAdminAccess 정책을 사용하는 관리자인 경우, 세션과 연결된 사용자 토큰이 있으면 인증 오류가 발생할 수 있습니다. 관리자는 세션과 관련된 사용자 토큰을 보유해서는 안 됩니다. 이 문제를 해결하려면 쿠키를 삭제하십시오.

계정이 다중 사용자 모드이고 AWSDeepracerDefaultMultiUserAccess 정책을 사용하는 레이서인 경우, 정책과 연결된 사용자 토큰이 없으면 인증 오류가 발생할 수 있습니다. 이 문제를 해결하려면 AWS DeepRacer를 계속 사용하기 전에 AWS 플레이어 프로필에 인증해야 합니다.

계정이 단일 사용자 모드이고 `AWSDeepRacerDefaultMultiUserAccess` 정책이 적용되는 레이서인 경우 인증 오류가 발생할 수 있습니다. 이 문제를 해결하려면 단일 사용자 모드에서 `AWSDeepRacerDefaultMultiUserAccess` 정책이 있는 사용자가 AWS DeepRacer를 사용할 수 없으므로 AWS 계정 관리자에게 문의하세요.

DeepRacer에서 작업을 수행할 권한이 없음

에서 작업을 수행할 권한이 없다는 AWS Management Console 메시지가 표시되면 관리자에게 문의하여 지원을 받아야 합니다. 관리자는 사용자 이름과 비밀번호를 제공한 사람입니다.

다음 예제 오류는 `mateojackson` IAM 사용자가 콘솔을 사용하여 가상 *`my-example-widget`* 리소스에 대한 세부 정보를 보려고 하지만 가상 `deepracer:`*`GetWidget`* 권한이 없을 때 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 deepracer:GetWidget on resource: my-example-widget
```

이 경우, Mateo는 *my-example-widget* 작업을 사용하여 `deepracer:`*GetWidget* 리소스에 액세스하도록 허용하는 정책을 업데이트하라고 관리자에게 요청합니다.

iam:PassRole을 수행하도록 인증되지 않음

`iam:PassRole` 작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 DeepRacer에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

일부 AWS 서비스 에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 기존 역할을 해당 서비스에 전달할 수 있습니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음 예제 오류는 `marymajor`(이)라는 IAM 사용자가 콘솔을 사용하여 DeepRacer에서 작업을 수행하려고 하는 경우에 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 수 있는 권한을 가지고 있지 않습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우, Mary가 `iam:PassRole` 작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 보안 인증 정보를 제공한 사람입니다.

액세스 키를 보아야 합니다.

IAM 사용자 액세스 키를 생성한 후에는 언제든지 액세스 키 ID를 볼 수 있습니다. 하지만 보안 액세스 키는 다시 볼 수 없습니다. 보안 액세스 키를 잃어버린 경우 새로운 액세스 키 페어를 생성해야 합니다.

액세스 키는 액세스 키 ID(예: `AKIAIOSFODNN7EXAMPLE`)와 보안 액세스 키(예: `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`)의 두 가지 부분으로 구성됩니다. 사용자 이름 및 암호와 같이 액세스 키 ID와 보안 액세스 키를 함께 사용하여 요청을 인증해야 합니다. 사용자 이름과 암호를 관리하는 것처럼 안전하게 액세스 키를 관리합니다.

> **Important**
>
> 정식 사용자 ID를 찾는 데 도움이 되더라도 액세스 키를 타사에 제공하지 마시기 바랍니다. 이렇게 하면 누군가에게에 대한 영구 액세스 권한을 부여할 수 있습니다 AWS 계정.

액세스 키 페어를 생성할 때는 액세스 키 ID와 보안 액세스 키를 안전한 위치에 저장하라는 메시지가 나타납니다. 보안 액세스 키는 생성할 때만 사용할 수 있습니다. 하지만 보안 액세스 키를 잃어버린 경우 새로운 액세스 키를 IAM 사용자에게 추가해야 합니다. 최대 두 개의 액세스 키를 가질 수 있습니다. 이미 두 개가 있는 경우 새로 생성하려면 먼저 키 페어 하나를 삭제해야 합니다. 지침을 보려면 IAM 사용 설명서의 액세스 키 관리를 참조하십시오.

관리자가 다른 사용자의 DeepRacer 액세스를 허용하려고 합니다

다른 사용자가 DeepRacer에 액세스하도록 허용하려면 액세스 권한이 필요한 사용자 또는 애플리케이션에 권한을 부여해야 합니다. AWS IAM Identity Center 를 사용하여 사용자 및 애플리케이션을 관리하는 경우 사용자 또는 그룹에 권한 세트를 할당하여 액세스 수준을 정의합니다. 권한 세트는 IAM 정책을 자동으로 생성하고 사용자 또는 애플리케이션과 연결된 IAM 역할에 할당합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 권한 세트를 참조하세요.

IAM Identity Center를 사용하지 않는 경우 액세스가 필요한 사용자 또는 애플리케이션에 대한 IAM 엔터티(사용자 또는 역할)를 생성해야 합니다. 그런 다음 DeepRacer에 대한 올바른 권한을 부여하는 정책을 엔터티에 연결해야 합니다. 권한이 부여되면 사용자 또는 애플리케이션 개발자에게 자격 증명을 제공합니다. 이들은 이 자격 증명을 사용하여 AWS에 액세스합니다. IAM 사용자, 그룹, 정책, 권한 생성에 대한 자세한 내용은 IAM 사용 설명서의 IAM ID 및 IAM 정책과 권한을 참조하세요.

내 외부의 사람이 내 DeepRacer 리소스 AWS 계정 에 액세스하도록 허용하고 싶습니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우, 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세히 알아보려면 다음을 참조하십시오.

- DeepRacer에서 이러한 기능을 지원하는지 여부를 알아보려면 AWS DeepRacer가 IAM과 작동하는 방법 단원을 참조하십시오.

- 소유 AWS 계정 한의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 소유 AWS 계정 한 다른의 IAM 사용자에게 액세스 권한 제공을 참조하세요.
- 타사에 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사 AWS 계정 소유에 대한 액세스 권한 제공을 AWS 계정참조하세요.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(ID 페더레이션)을 참조하세요.
- 크로스 계정 액세스에 대한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 크로스 계정 리소스 액세스를 참조하세요.

태그 지정

태그는 사용자 또는가 AWS 리소스에 AWS 할당하는 사용자 지정 속성 레이블입니다. 각 AWS 태그에는 두 부분이 있습니다.

- 태그 키(예: `companyname`, `costcenter`, `environment`, `project`, 또는 `secret`) 태그 키는 대소문자를 구별합니다.
- 태그 값으로 알려진 선택적 필드 태그 값을 생략하는 것은 빈 문자열을 사용하는 것과 같습니다. 태그 키처럼 태그 값은 대/소문자를 구별합니다.

태그 키와 태그 값을 합해서 키 값 페어라고 합니다.

AWS DeepRacer 서비스에서는 자동차, RL 모델 및 커뮤니티 레이스 리더보드에 태그를 할당할 수 있습니다. 이러한 AWS 리소스와 태깅을 지원하는 기타 리소스에 태그를 지정하여 리소스가 관련이 있음을 나타냅니다. 태그로 모델 및 리더보드를 식별 및 구성하는 것 외에도 태그를 사용하여 비용 할당을 추적하고 IAM 정책에서 리소스를 보고 상호 작용할 수 있는 사용자를 제어할 수 있습니다. AWS DeepRacer 콘솔 또는 AWS CLI 를 사용하여 태그를 추가, 관리 및 제거합니다.

태그 사용에 대한 자세한 내용은 태그 지정 모범 사례 백서를 참조하십시오.

비용 할당을 추적하기 위한 태그

AWS Cost Explorer와 비용 및 사용 보고서는 AWS 비용을 태그별로 분류하는 기능을 지원합니다. `cost center`, `businessunit`, 또는 `project`와 같은 비즈니스 태그를 사용하여 AWS 비용을 조직의 일반적인 재무 보고 카테고리와 연결할 수 있습니다. 그러나 비용 할당 보고서에는 특정 애플리케이션, 환경 또는 규정 준수 프로그램과 같은 기술 또는 보안 범주와 비용을 쉽게 연결할 수 있는 모든 태그가 포함될 수 있습니다. 조직의 관리 계정과 조직의 멤버가 아닌 단일 계정만 Billing and Cost Management 콘솔의 비용 할당 태그 관리자에 액세스할 수 있습니다. 태그를 사용하여 비용할당을 추적하는 방법에 대한 자세한 내용은 사용자 정의 비용 할당 태그를 참조하십시오.

액세스를 관리하기 위한 태그

또한 IAM 사용자 및 역할을 태그하여 모델 및 커뮤니티 레이스 리더보드에 대한 액세스를 관리할 수 있습니다. IAM 사용자 및 역할을 태그 지정하는 방법을 알아보려면 IAM 사용자 및 역할을 태그 지정하는 방법을 참조하십시오. 보안 주체 태그가 있는 IAM 역할이 일치하는 태그가 있는 리소스에 액세스할 수 있도록 허용하는 정책을 생성하고 테스트하기 위한 자습서를 보려면 IAM 자습서: 태그를 기반으로 AWS 리소스에 액세스할 수 있는 권한 정의를 참조하세요. 태그를 사용하여 태그 지정을 지원하는

AWS 리소스에 대한 액세스를 제어하는 방법에 대한 자세한 내용은 리소스 태그를 사용하여 AWS 리소스에 대한 액세스 제어를 참조하세요.

주제

- 새 리소스에 대한 태그 추가, 보기, 편집
- 기존 리소스에 대한 태그 추가, 보기 및 편집

새 리소스에 대한 태그 추가, 보기, 편집

새 자동차, RL 모델 또는 커뮤니티 레이스 리더보드에 태그를 추가하면 이러한 리소스를 식별, 구성, 비용 할당 추적 및 관리에 도움이 될 수 있습니다. 모델이나 리더보드에 하나 이상의 태그(키-값 페어)를 추가합니다. 각 리소스에 대해 각 태그 키는 고유해야 하며 각 태그 키는 하나의 값만 가질 수 있지만 하나의 리소스는 최대 50개까지의 태그를 가질 수 있습니다.

AWS DeepRacer 콘솔에서 한 번에 한 리소스씩 태그를 생성 및 적용하거나 태그 편집기를 사용하여 여러 리소스를 한 번에 추가, 편집 또는 삭제할 수 있습니다.

> **Important**
>
> RL 모델 또는 커뮤니티 레이스 리더보드의 태그를 편집하면 해당 리소스에 대한 액세스에 영향을 미칠 수 있습니다. 태그의 이름(키) 또는 값을 편집하기 전에 그러한 리소스에 대한 액세스를 제어하는 태그의 키 또는 값을 사용할 수도 있는 모든 IAM 정책을 검토하십시오.

새 RL 모델에 대한 태그 추가, 보기, 편집하기

AWS DeepRacer 콘솔을 사용하여 새 RL 모델에 태그를 추가하고, 보고, 편집할 수 있습니다.

1. 모델에서 모델 생성을 선택합니다.
2. 모델 생성 페이지에서 훈련 세부 정보를 입력한 후 태그 제목을 확장합니다.
3. 태그 제목 아래에서 새 태그 추가를 선택합니다.
4. 키에 태그 이름을 입력합니다. 값에 태그의 선택적 값을 추가할 수 있습니다. 태그 이름 지정에 대한 자세한 내용은 태깅 모범 사례 백서의 태그 및 리소스 이름 지정 모범 사례 항목을 참조하십시오.
5. (선택 사항) 다른 태그를 추가하려면 다시 새 태그 추가를 선택합니다.
6. (선택 사항) 개별 키나 값을 제거하려면 옆에 있는 X를 선택합니다.
7. (선택 사항) 키-값 페어를 제거하려면 제거를 선택합니다.
8. 태그 추가를 마치면 환경 시뮬레이션에서 트랙을 선택하고 다음을 선택합니다.

훈련용으로 새 모델에 태그를 지정하고 제출한 후에는 페이지 하단의 태그 제목에서 훈련 및 평가 중 또는 이후에 해당 태그를 관리할 수 있습니다.

1. 태그 관리를 선택합니다.
2. 태그 관리 팝업 상자에서 제거하려는 태그 옆의 제거 버튼을 선택하여 생성한 태그를 제거하거나 새 태그 추가를 선택하여 새 태그를 추가할 수 있습니다.
3. 새 태그를 추가하는 경우 키에 해당 태그의 이름을 입력합니다. 값에 태그의 선택적 값을 추가할 수 있습니다. 태그 이름 지정에 대한 자세한 내용은 태깅 모범 사례 백서의 태그 및 리소스 이름 지정 모범 사례 항목을 참조하십시오.
4. 태그 제거 및 추가를 마쳤으면 제출을 선택합니다.

새 커뮤니티 레이스 리더보드에 태그를 추가, 보고, 편집하려면

AWS DeepRacer 콘솔을 사용하여 새 커뮤니티 레이스 리더보드에 태그를 추가, 보고, 편집할 수 있습니다.

1. 커뮤니티 레이스에서 레이스 만들기를 선택합니다.
2. 레이스 세부 정보 페이지에서 태그 제목을 확장하십시오.
3. 태그 제목 아래에서 새 태그 추가를 선택합니다.
4. 키에 태그 이름을 입력합니다. 값에 태그의 선택적 값을 추가할 수 있습니다. 태그 이름 지정에 대한 자세한 내용은 태깅 모범 사례 백서의 태그 및 리소스 이름 지정 모범 사례 항목을 참조하십시오.
5. (선택 사항) 다른 태그를 추가하려면 다시 새 태그 추가를 선택합니다.
6. (선택 사항) 개별 키나 값을 제거하려면 옆에 있는 X를 선택합니다.
7. (선택 사항) 키-값 페어를 제거하려면 제거를 선택합니다.
8. 태그 추가를 마치면 환경 시뮬레이션에서 트랙을 선택하고 다음을 선택합니다.

기존 리소스에 대한 태그 추가, 보기 및 편집

기존 AWS DeepRacer RL 모델 또는 커뮤니티 레이스 리더보드에 태그를 추가하면 비용 할당을 식별, 구성, 추적하고 이러한 리소스에 대한 액세스를 관리하는 데 도움이 될 수 있습니다. 모델이나 리더보드에 하나 이상의 태그(키-값 페어)를 추가합니다. 각 리소스에 대해 각 태그 키는 고유해야 하며 각 태그 키는 하나의 값만 가질 수 있지만 하나의 리소스는 최대 50개까지의 태그를 가질 수 있습니다.

AWS DeepRacer 콘솔에서 한 번에 한 리소스씩 태그를 생성 및 적용하거나 태그 편집기를 사용하여 여러 리소스를 한 번에 추가, 편집 또는 삭제할 수 있습니다.

> **Important**
>
> RL 모델 또는 커뮤니티 레이스 리더보드의 태그를 편집하면 해당 리소스에 대한 액세스에 영향을 미칠 수 있습니다. 태그의 이름(키) 또는 값을 편집하기 전에 그러한 리소스에 대한 액세스를 제어하는 태그의 키 또는 값을 사용할 수도 있는 모든 IAM 정책을 검토하십시오.

기존 RL 모델의 태그 추가, 보기, 편집하기

AWS DeepRacer 콘솔을 사용하여 기존 RL 모델의 태그를 추가, 보기 또는 편집할 수 있습니다.

1. 모델의 목록에서 모델 이름을 선택하여 모델을 선택합니다.
2. 작업을 선택합니다.
3. 드롭다운 목록에서 태그 관리를 선택합니다.
4. 태그 관리 팝업 상자에서 태그를 보거나 추가 또는 제거할 수 있습니다.
   a. 태그를 추가하려면 태그 추가를 선택합니다. 키에 태그 이름을 입력합니다. 값에 태그의 선택적 값을 추가할 수 있습니다. 태그 이름 지정에 대한 자세한 내용은 태깅 모범 사례 백서의 태그 및 리소스 이름 지정 모범 사례 항목을 참조하십시오.
   b. 다른 태그를 추가하려면 새 태그 추가를 다시 선택합니다.
   c. 개별 키나 값을 제거하려면 해당 키 또는 값 옆에 있는 X를 선택합니다.
   d. 키-값 페어를 제거하려면 제거를 선택합니다.
5. 태그 보기, 추가, 제거를 마쳤으면 제출을 선택합니다.

기존 커뮤니티 레이스 리더보드에 태그를 추가, 보고, 편집하려면

1. 커뮤니티 레이스에서 레이스 관리를 선택합니다.
2. 레이스 관리 페이지에서 레이스를 선택합니다.
3. 작업을 선택합니다.
4. 드롭다운 목록에서 태그 관리를 선택합니다.
5. 태그 관리 팝업 상자에서 태그를 보거나 추가 또는 제거할 수 있습니다.
   a. 태그를 추가하려면 태그 추가를 선택합니다. 키에 태그 이름을 입력합니다. 값에 태그의 선택적 값을 추가할 수 있습니다. 태그 이름 지정에 대한 자세한 내용은 태깅 모범 사례 백서의 태그 및 리소스 이름 지정 모범 사례 항목을 참조하십시오.
   b. 다른 태그를 추가하려면 새 태그 추가를 다시 선택합니다.

   c. 개별 키나 값을 제거하려면 해당 키 또는 값 옆에 있는 X를 선택합니다.
   d. 키-값 페어를 제거하려면 제거를 선택합니다.
6. 태그 보기, 추가, 제거를 마쳤으면 제출을 선택합니다.

일반적인 AWS DeepRacer 문제 해결

여기에는 FAQ 및 최신 버그 수정 사항에 대한 문제 해결 팁이 나와 있습니다.

주제

- 일반적인 AWS DeepRacer 라이브 문제를 해결하는 방법
- 내 컴퓨터와 차량 간 USB 연결을 통해 디바이스 콘솔에 연결할 수 없는 이유는 무엇입니까?
- AWS DeepRacer 컴퓨팅 모듈 전원은 배터리에서 콘센트로 어떻게 전환합니까?
- USB 플래시 드라이브를 사용하여 Wi-Fi 네트워크에 AWS DeepRacer을 연결하는 방법
- AWS DeepRacer 주행 모듈 배터리를 충전하는 방법
- AWS DeepRacer 컴퓨팅 모듈 배터리를 충전하는 방법
- 배터리가 충전되었지만 AWS DeepRacer 차량이 움직이지 않음
- AWS DeepRacer 차량 배터리 잠금 문제를 해결합니다.
- LiDAR 센서를 설치할 때 Dell 배터리 커넥터 케이블을 감싸는 방법
- 차량의 Wi-Fi 연결을 유지하는 방법
- AWS DeepRacer 디바이스의 MAC 주소를 가져오는 방법
- AWS DeepRacer 디바이스 콘솔의 기본 암호를 복구하는 방법
- AWS DeepRacer 디바이스를 수동으로 업데이트하는 방법
- 일반적인 AWS DeepRacer 운영 문제를 진단하고 해결하는 방법

일반적인 AWS DeepRacer 라이브 문제를 해결하는 방법

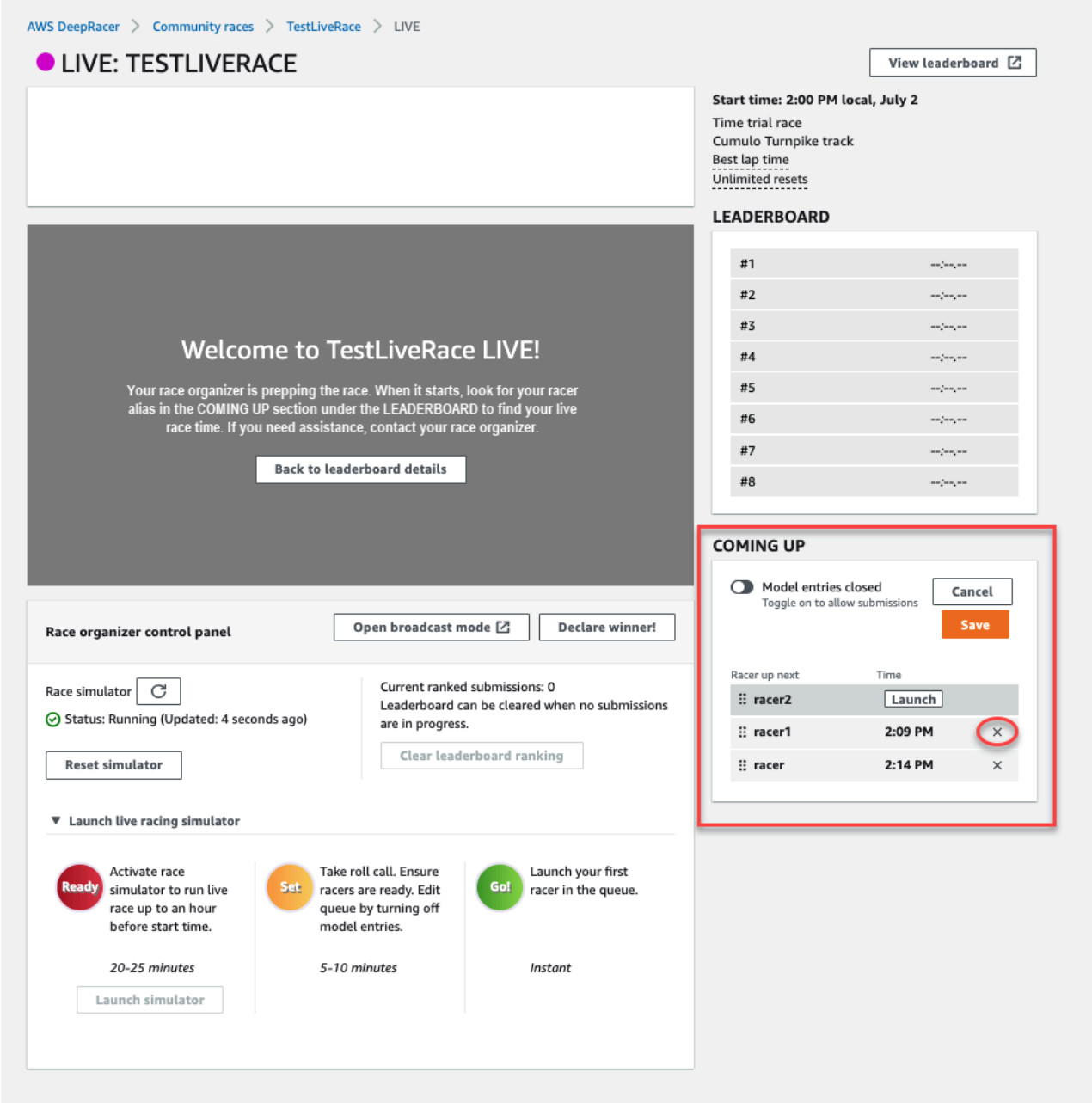
라이브 레이스 페이지에서 레이스 비디오를 볼 수 없습니다.

- 가상 프라이빗 네트워크(VPN)를 사용 중인 경우 레이싱 이벤트 중에 연결이 끊겼는지 확인하십시오.
- 디바이스에 광고 차단기가 설치되어 있는 경우 레이싱 이벤트 중에 연결이 끊겼는지 확인하십시오.
- 홈 네트워크에서 광고 차단기를 실행하는 경우 레이싱 이벤트 중에 연결이 끊겼는지 확인하십시오.

레이스 대기열에 있는 레이서 이름이 빨간색으로 표시됩니다.

LIVE: <Your Race Name> 페이지의 다음 섹션에서 레이서 이름이 빨간색으로 강조 표시되면 해당 레이서의 모델 제출에 문제가 있음을 의미합니다.

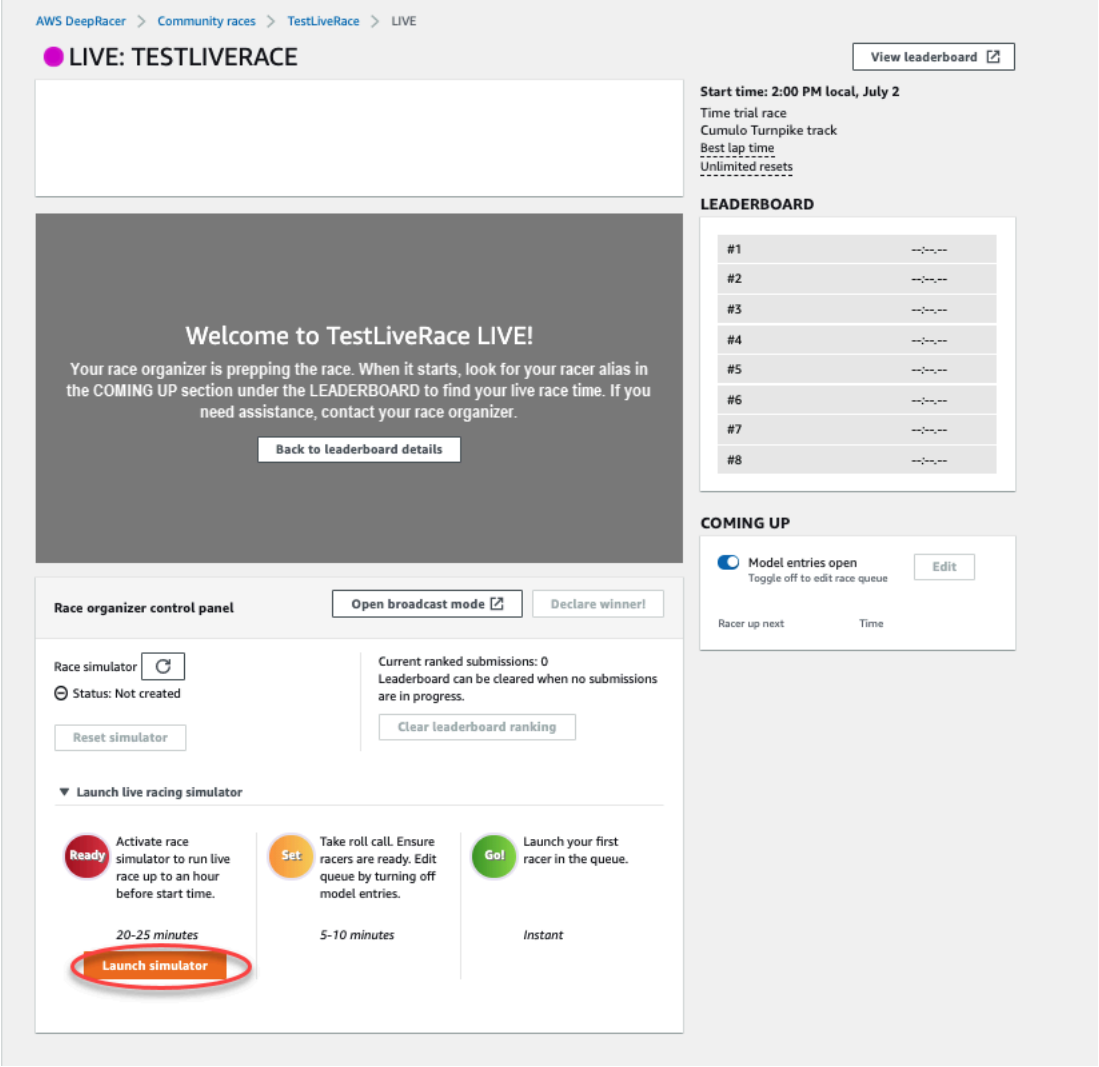
- 레이스 주최자인 경우 LIVE: <Your Race Name> 페이지의 다음 섹션에서 편집을 선택하여 해당 레이서의 이름이 포함된 행에서 X를 선택하여 레이서가 제출한 모델을 삭제하십시오. 그 후 저장을 선택합니다. 대기열 재정렬에 대한 도움말은 the section called "라이브 레이스를 진행해 보십시오."의 11단계를 참조하십시오.



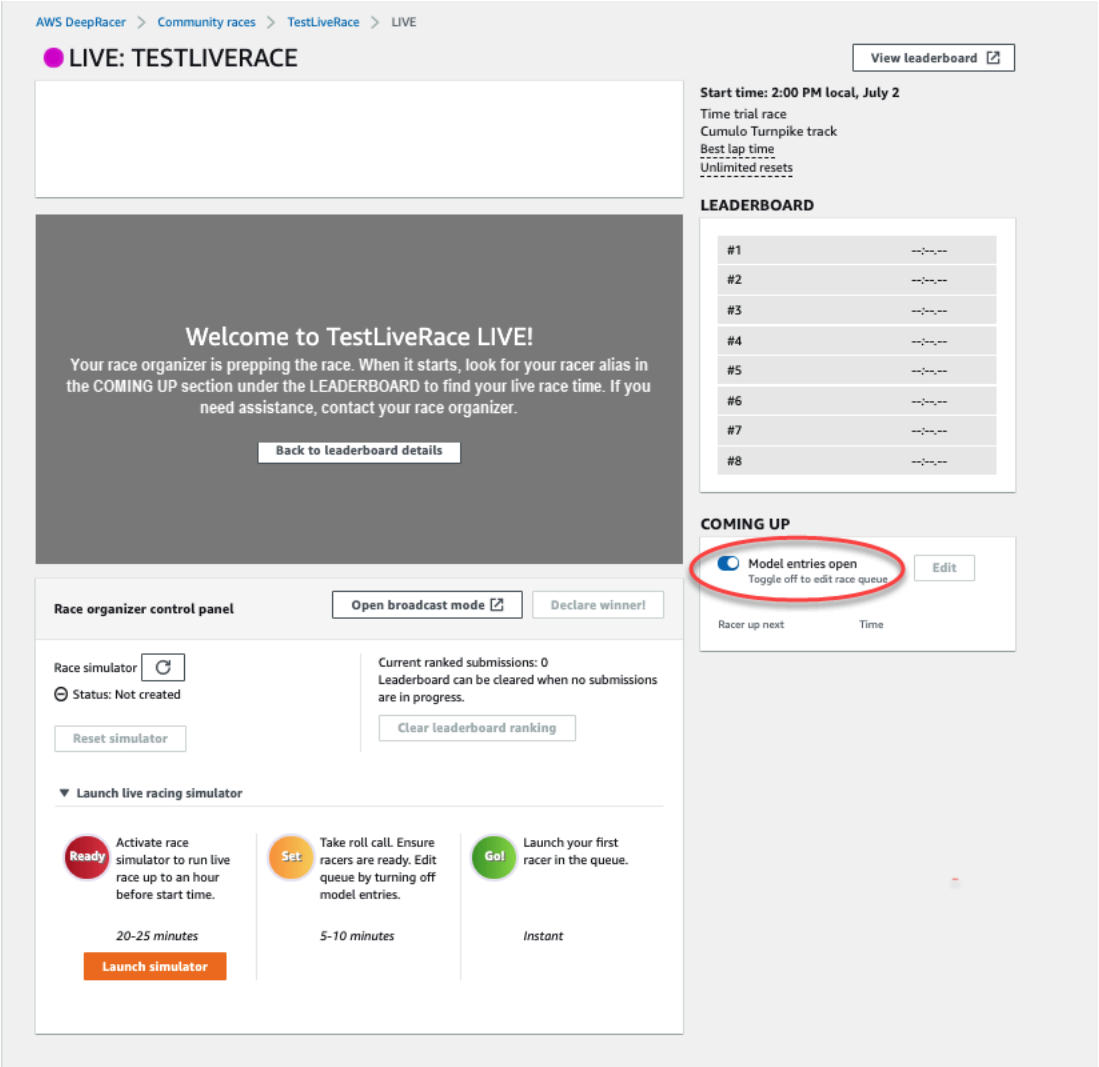
- 레이서라면 모델을 레이스에 다시 제출하십시오. the section called “라이브 레이스를 진행해 보십시오.”로 이동하여 라이브 레이스 참가하기를 선택하여 도움을 받으십시오.

라이브 레이스를 진행 중인데 레이서를 런칭할 수 없어요

- LIVE: <Your Race Name> 페이지의 라이브 레이싱 시뮬레이터 실행 섹션에서 시뮬레이터 실행를 선택했는지 확인하십시오. 도움이 더 필요하면 the section called “라이브 레이스를 진행해 보십시오.”의 2단계를 참조하십시오.



- LIVE: <Your Race Name> 페이지의 COMING UP에서 제출물을 마감하기 위해 모델 항목 열기를 해제했는지 확인하십시오. 도움이 더 필요하면 the section called “라이브 레이스를 진행해 보십시오.”의 3단계를 참조하십시오.



Chrome 또는 Firefox 브라우저를 사용하고 있는데 라이브 레이스를 보는 데 여전히 문제가 있습니다.

- 최신 버전의 Chrome 또는 Firefox 브라우저가 적용되도록 합니다. 그렇지 않은 경우 브라우저를 최신 버전으로 업데이트하고 레이스를 다시 시청해 보십시오.
- 가상 프라이빗 네트워크(VPN)를 사용 중인 경우 연결이 끊겼는지 확인하십시오.
- 디바이스에서 광고 차단기가 실행되는 경우 레이싱 이벤트 중에 연결이 끊겼는지 확인하십시오.
- 홈 네트워크에서 광고 차단기를 실행하는 경우 레이싱 이벤트 중에 연결이 끊겼는지 확인하십시오.
- 인터넷 브라우저에서 WebRTC가 꺼져 있는 경우, 레이싱 이벤트 중에 WebRTC를 켜십시오.

내 컴퓨터와 차량 간 USB 연결을 통해 디바이스 콘솔에 연결할 수 없는 이유는 무엇입니까?

차량을 처음 설정할 때 micro-USB/USB 케이블(USB를 USB-A라고도 함)을 통해 AWS DeepRacer 차량을 컴퓨터에 연결한 후에 디바이스 콘솔을 열지 못할 수 있습니다(차량에서 호스팅되는 디바이스 웹 서버(`https://deepracer.aws`)라고도 함).

다양한 원인이 있을 수 있습니다. 일반적으로 이 문제는 다음과 같은 간단한 해결 방법을 통해 해결할 수 있습니다.

디바이스의 USB 오버 이더넷 네트워크를 활성화하려면

1. 컴퓨터에서 Wi-Fi를 끄고 연결된 이더넷 케이블을 분리합니다.
2. 차량에서 재설정 버튼을 눌러 디바이스를 재부팅합니다.
3. 컴퓨터의 웹 브라우저에서 `https://deepracer.aws`로 이동하여 디바이스 콘솔을 엽니다.

위의 절차로 해결할 수 없는 경우 컴퓨터가 디바이스의 네트워크(네트워크 이름: `Deepracer`)에 연결되도록 컴퓨터의 네트워크 기본 설정이 제대로 구성되어 있는지 확인할 수 있습니다. 이렇게 하려면 다음 절차의 단계를 따릅니다.

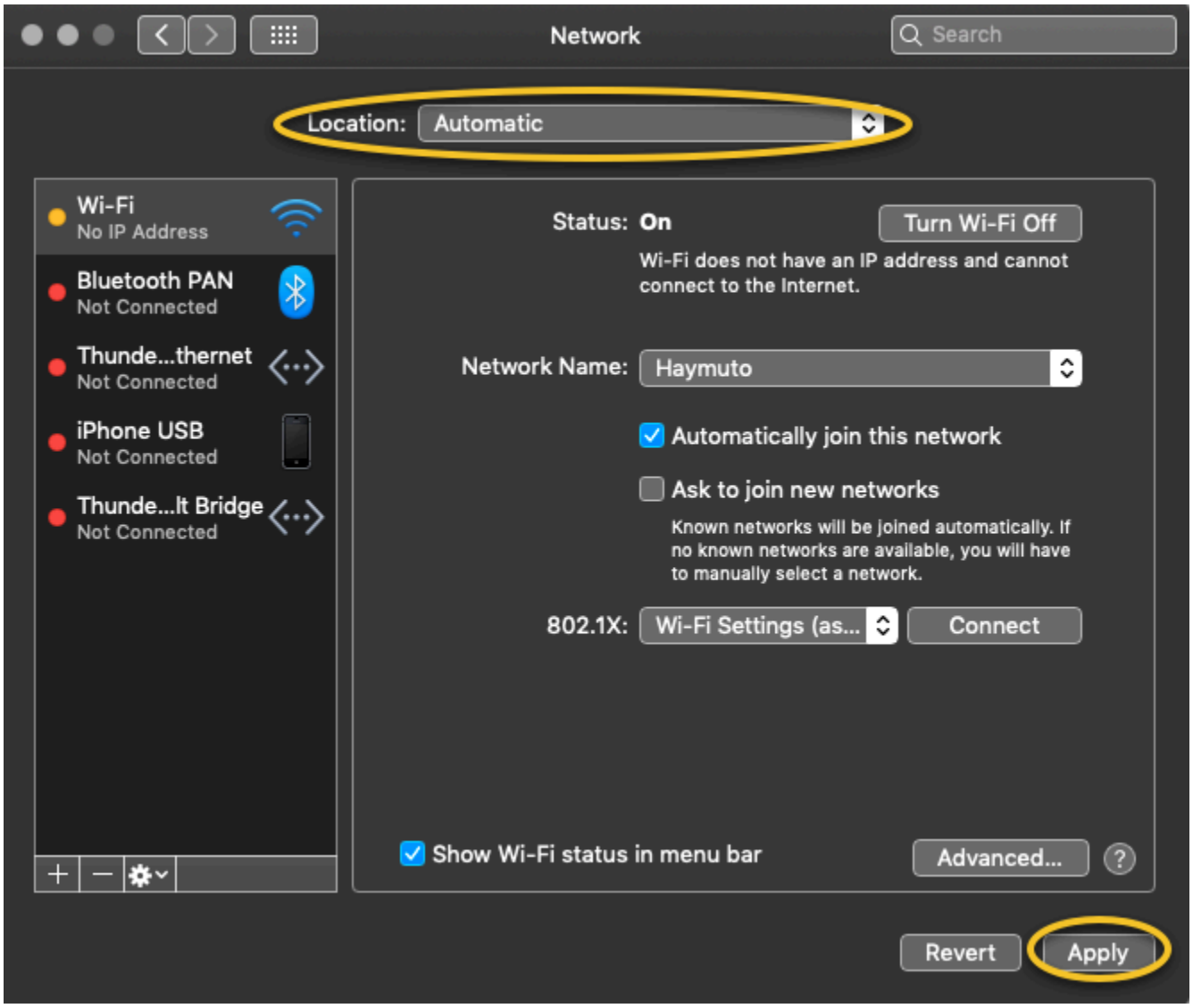
> **Note**
>
> 아래 지침에서는 MacOS 컴퓨터를 사용한다고 가정합니다. 다른 컴퓨터 시스템의 경우 해당 운영 체제에 대한 네트워크 기본 설정 설명서를 참조하고 아래 지침을 일반적인 가이드로 활용하십시오.

MacOS 컴퓨터에서 디바이스의 USB 오버 이더넷 네트워크를 활성화하려면

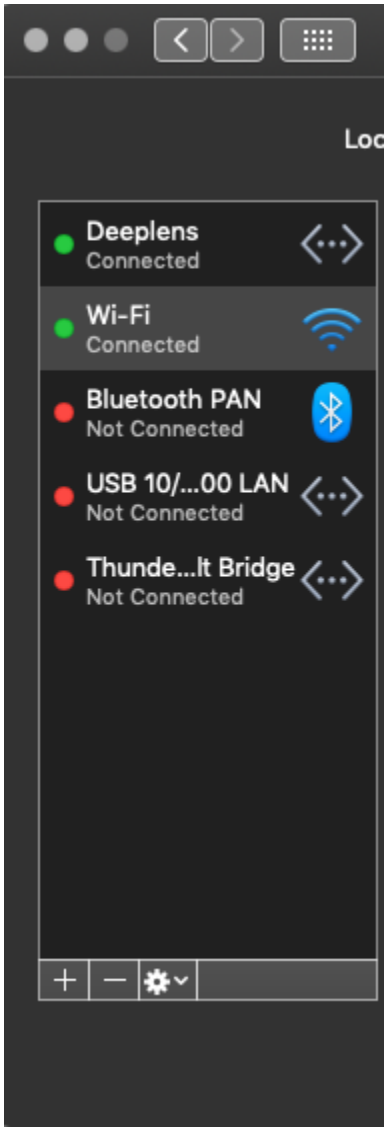
1. 네트워크 아이콘(디스플레이 오른쪽 상단)을 선택하여 네트워크 기본 설정를 엽니다.

   또는 Command+space를 선택하고 네트워크를 입력한 후 네트워크 시스템 기본 설정를 선택합니다.

2. Deepracer가 연결됨로 나열되어 있는지 확인합니다. DeepRacer가 나열되어 있지만 연결되지 않은 경우 마이크로 USB/USB 케이블이 차량과 컴퓨터 간에 제대로 연결되어 있는지 확인합니다.
3. Deepracer 네트워크가 목록에 없거나 목록에 있지만 USB 케이블을 연결했을 때 연결되지 않은 경우, 위치 환경설정에서 자동을 선택한 다음 적용을 선택합니다.



4. AWS DeepRacer 네트워크가 연결됨로 실행 중인지 확인합니다.



5. 컴퓨터가 Deepracer 네트워크에 연결되면 브라우저의 `https://deepracer.aws` 페이지를 새로고침하고 시작 안내서의 나머지 Wi-Fi에 연결 지침을 계속 진행하십시오.
6. Deepracer 네트워크가 연결되지 않은 경우 AWS DeepRacer 차량에서 컴퓨터를 연결 해제한 다음 다시 연결합니다. Deepracer 네트워크가 연결됨 상태가 되면 시작 안내서 지침을 계속 진행합니다.
7. 디바이스의 Deepracer 네트워크가 아직 연결되지 않은 경우 컴퓨터 및 AWS DeepRacer 차량을 재부팅하고 필요한 경우 이 절차의 1단계부터 반복합니다.

위의 문제 해결로도 문제를 해결할 수 없는 경우, 디바이스 인증서가 손상되었을 수 있습니다. 아래 단계에 따라 AWS DeepRacer 차량에 대한 새로운 인증서를 생성하여 손상된 파일을 복구합니다.

AWS DeepRacer 차량에 대한 새로운 인증서를 생성하려면

1. micro-USB/USB 케이블을 분리하여 컴퓨터와 AWS DeepRacer 차량 간에 USB 연결을 종료합니다.
2. AWS DeepRacer 차량을 모니터(HDMI - HDMI 케이블 사용)와 USB 키보드 및 마우스에 연결합니다.
3. AWS DeepRacer 운영 체제에 로그인합니다. 디바이스 운영 체제에 처음 로그인하는 경우, 메시지가 표시되면 암호에 `deepracer`를 사용하고 필요한 경우 암호를 변경합니다. 후속 로그인 시 업데이트된 암호를 사용합니다.
4. 터미널 창을 열고 다음 셸 명령을 입력합니다. 데스크탑의 애플리케이션 -> 시스템 도구에서 터미널 단축키를 선택하여 터미널 창을 열 수 있습니다. 또는 파일 브라우저를 사용하고 `/usr/bin` 폴더로 이동한 후 gnome-terminal을 선택하여 열 수도 있습니다.

   ```
   sudo /opt/aws/deepracer/nginx/nginx_install_certs.sh && sudo reboot
   ```

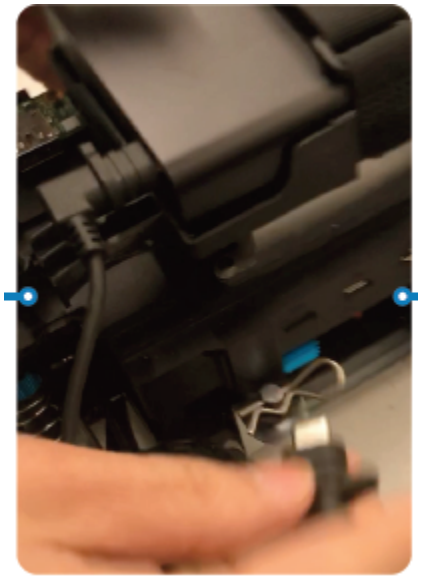
   메시지가 표시되면 이전 단계에서 사용했거나 업데이트한 암호를 입력합니다.

   위의 명령을 사용하면 새 인증서가 설치되고 디바이스가 재부팅됩니다. 또한 디바이스 콘솔의 암호를 AWS DeepRacer 차량 하단에 인쇄된 기본값으로 되돌립니다.
5. 차량에서 모니터, 키보드 및 마우스를 연결 해제하고 micro-USB/USB 케이블을 사용하여 컴퓨터에 다시 연결합니다.
6. 이 주제의 두 번째 절차에 따라 컴퓨터가 디바이스 네트워크에 실제로 연결되어 있는지 확인한 다음 디바이스 콘솔(`https://deepracer.aws`)을 다시 열기 전에 시작 안내서의 Wi-Fi에 연결 지침을 계속 진행하십시오.

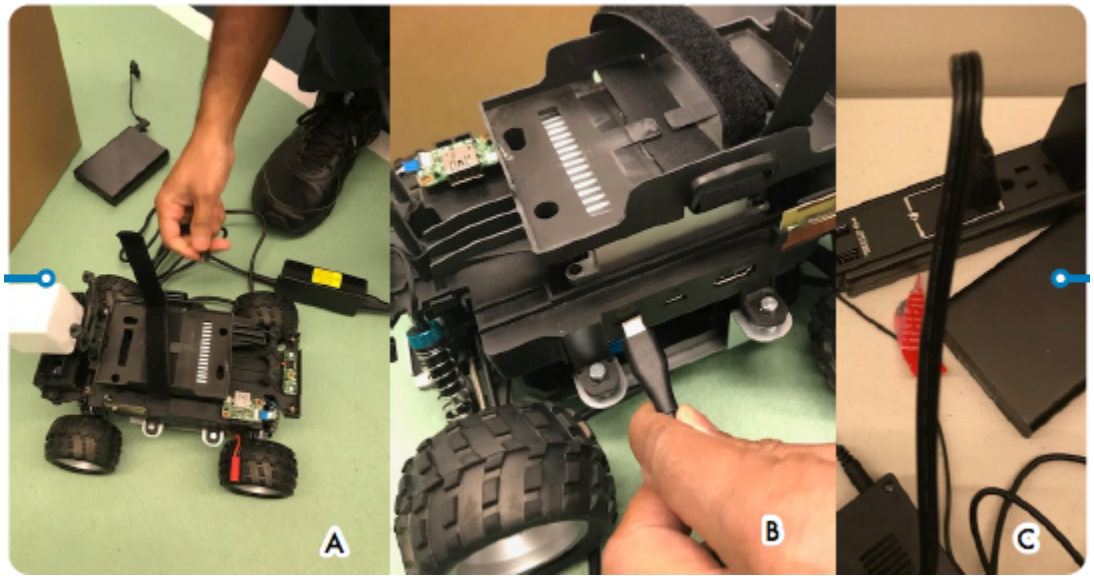
AWS DeepRacer 컴퓨팅 모듈 전원은 배터리에서 콘센트로 어떻게 전환합니까?

AWS DeepRacer를 처음으로 설정할 때 컴퓨팅 모듈 배터리 잔량이 부족한 경우 아래 단계에 따라 컴퓨팅 전원을 배터리에서 콘센트로 전환합니다.

1. 차량의 컴퓨팅 전원 포트에서 USB-C 케이블을 뽑습니다.



2. 컴퓨터 모듈 전원 어댑터(A)에 AC 전원 코드와 USB-C 케이블을 연결합니다. 전원 코드를 콘센트(C)에 꽂고, USB-C 케이블을 차량의 컴퓨터 모듈 전원 포트(B)에 꽂습니다.



USB 플래시 드라이브를 사용하여 Wi-Fi 네트워크에 AWS DeepRacer을 연결하는 방법

USB 플래시 드라이브를 사용하여 AWS DeepRacer 차량을 가정 또는 사무실 Wi-Fi 네트워크에 연결하려면 다음이 필요합니다.

- USB 플래시 드라이브
- 연결하려는 Wi-Fi 네트워크의 이름(SSID) 및 암호

> **Note**
>
> AWS DeepRacer에서는 로그인할 때 활성 캡처 확인을 요구하는 Wi-Fi 네트워크를 지원하지 않습니다.

USB 플래시 드라이브를 사용하여 AWS DeepRacer 차량을 Wi-Fi 네트워크에 연결하려면

1. 컴퓨터에 USB 플래시 드라이브를 꽂습니다.
2. 컴퓨터에서 웹 브라우저를 열고 https://aws.amazon.com/deepracer/usbwifi로 이동합니다. 이 링크는 GitHub에서 호스팅되는 `wifi-creds.txt`라는 텍스트 파일을 엽니다.

40 lines (39 sloc) | 3.25 KB

```
##################################################################################
#                     Connect the AWS DeepRacer vehicle to Wi-Fi                 #
# File name: wifi-creds.txt                                                      #
#                                                                                #
# To connect the AWS DeepRacer vehicle to Wi-Fi, type your Wi-Fi name (SSID) and #
# password in the appropriate field at the end of this file. Both values are case #
# sensitive.                                                                     #
#                                                                                #
# For example:                                                                   #
#      ssid: 'Your-WiFi 100'                                                     #
#      password: 'Passwd1234'                                                    #
#                                                                                #
```

3. USB 플래시 드라이브에 `wifi-creds.txt`를 저장합니다. 사용하는 웹 브라우저에 따라 텍스트 파일이 컴퓨터에 다운로드되어 기본 코드 편집기에서 자동으로 열릴 수 있습니다. `wifi-creds.txt`가 자동으로 다운로드되지 않으면 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 다른 이름으로 저장을 선택하여 텍스트 파일을 USB 플래시 드라이브에 저장합니다.

> **Warning**
>
> 파일 이름을 변경하지 마십시오.

4. `wifi-creds.txt`이 열려 있지 않으면 코드 편집기에서 일반 텍스트 모드로 여십시오. 일부 텍스트 편집기는 파일 유형이 지정되지 않은 경우 기본적으로 일반 텍스트(.txt) 대신 리치 텍스트(.rtf)를 사용하므로 파일을 편집하는 데 문제가 있는 경우 설정을 확인하십시오. Windows를 사용하는 경우 무료로 다운로드할 수 있는 Sublime Text 응용 프로그램을 사용하여 파일을 열거나, Mac을 사용하는 경우 대부분의 Mac 디바이스에 사전 설치되어 있는 기본 일반 텍스트로 설정된 TextEdit 응용 프로그램을 사용해 볼 수도 있습니다.

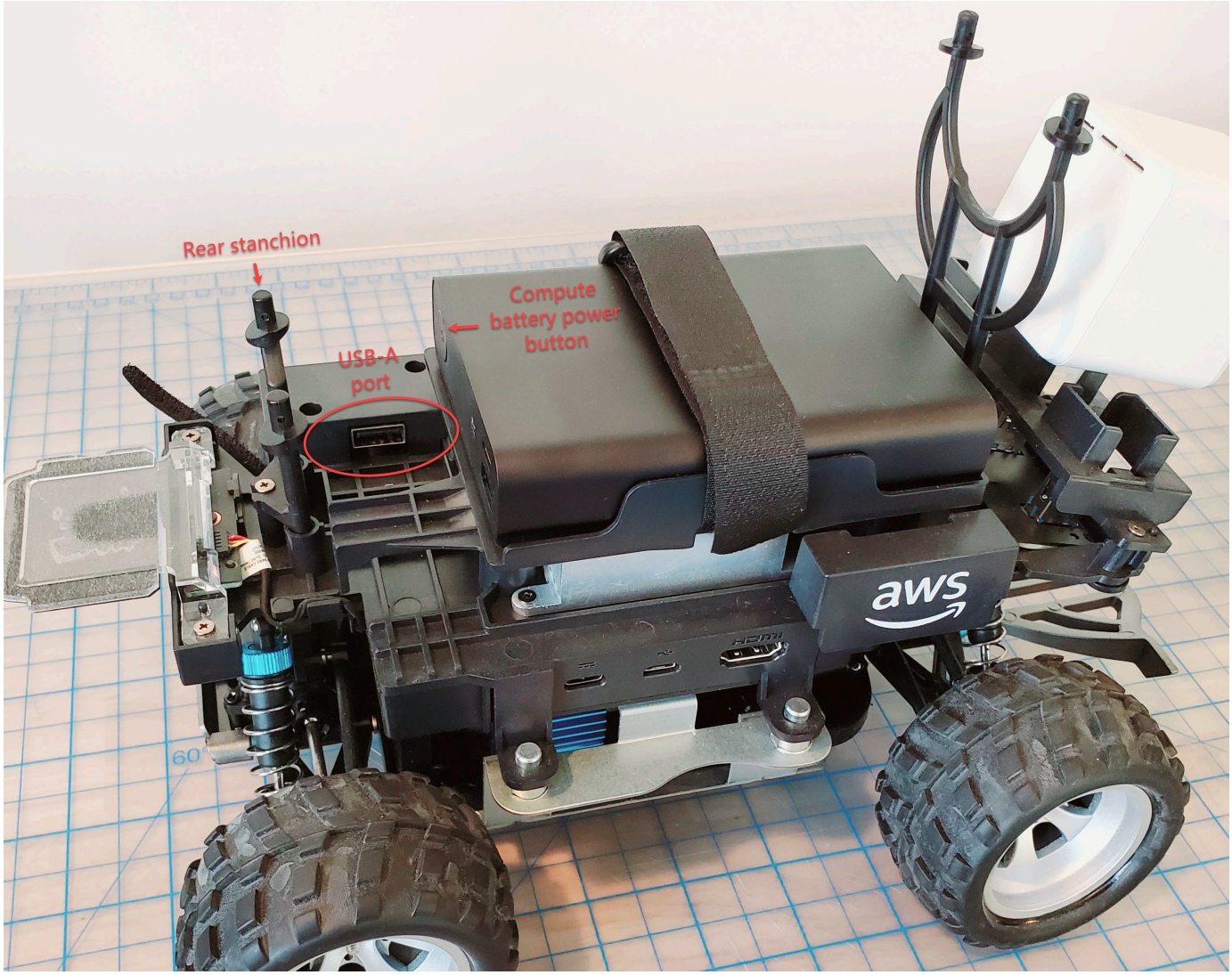
5. 파일 하단의 작은 따옴표 사이에 사용할 Wi-Fi 네트워크의 이름(SSID)과 암호를 입력합니다. SSID는 “Service Set Identifier“의 약어입니다. Wi-Fi 네트워크 이름에 대한 기술 용어입니다.

> **Note**
>
> 네트워크 이름(SSID) 또는 암호에 Your-Wi-Fi 100와 같은 공백이 포함된 경우 공백을 포함하여 작은 따옴표(‘’)안에 정확하게 이름을 입력합니다. 공백이 없으면 따옴표 사용은 선택 사항입니다. 예를 들어 Wi-Fi 비밀번호인 Passwd1234에는 공백이 없으므로 작은따옴표를 사용하는 것은 가능하지만 필수는 아닙니다. SSID와 암호는 모두 대/소문자를 구분합니다.

```
# If you have validated the Wi-Fi credentials but the Wi-Fi LED doesn't          #
# turn solid blue, try restarting the vehicle by pressing the reset button.       #
# When the power LED turns blue, plug the USB drive in again.                     #
#                                                                                 #
# To finish setting up, follow the instructions on https://docs.aws.amazon.com/   #
# deepracer/latest/developerguide/deepracer-troubleshooting-wifi-connection-first #
# -time.                                                                          #
###################################################################################

# Enter your Wi-Fi name (SSID) and password:
ssid: ''
password: ''
```

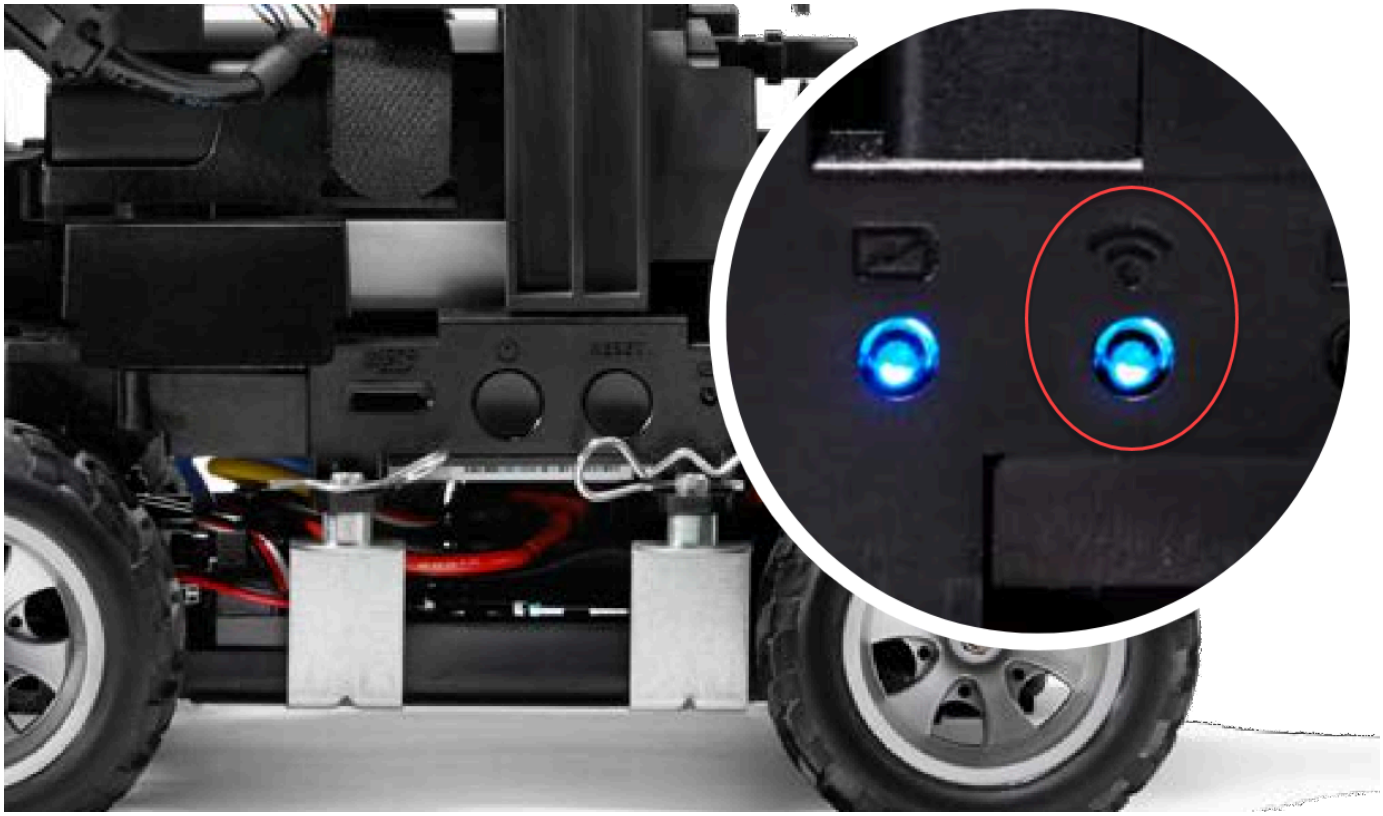
6. USB 플래시 드라이브에 파일을 저장합니다.

7. 컴퓨터에서 USB 드라이브를 꺼낸 다음, AWS DeepRacer 차량 후면의 컴퓨터 배터리 전원 버튼과 후면 받침대 사이에 있는 USB-A 포트에 연결합니다.



8. AWS DeepRacer이 켜져 있는지 확인합니다.
9. 차량의 Wi-Fi LED를 확인합니다. 깜박인 다음 흰색에서 파란색으로 변경되면 차량이 Wi-Fi 네트워크에 연결된 것입니다. USB 드라이브를 분리하고 11단계로 건너뜁니다.

> **Note**
>
> 차량을 Wi-Fi 네트워크에 연결하기 전에 USB 드라이브를 차량에 연결한 경우, 사용 가능한 Wi-Fi 네트워크 목록이 플래시 드라이브 `wifi-creds.txt` 파일에 자동으로 표시됩니다. 파운드 기호를 제거하여 연결하려는 네트워크의 주석을 제거합니다.



10. 차량 Wi-Fi LED가 깜박인 다음 빨간색으로 바뀌면 차량에서 USB 드라이브를 뽑고 다시 컴퓨터에 꽂습니다. 텍스트 파일에 입력한 Wi-Fi 이름과 암호에 오타, 간격 오류, 잘못된 문장 대소문자 또는 작은따옴표 누락이나 오용이 없는지 확인합니다. 실수를 수정하고 파일을 다시 저장한 다음 7~9 단계를 반복합니다.
11. 차량 Wi-Fi LED가 파란색으로 바뀌면 차량에서 USB 드라이브를 뽑고 컴퓨터에 꽂습니다.
12. `wifi-creds.txt` 파일을 엽니다. 텍스트 파일 하단에서 차량의 IP 주소를 찾아 복사합니다.
13. 컴퓨터가 차량과 동일한 네트워크에 있는지 확인한 다음 IP 주소를 웹 브라우저에 붙여넣습니다.

> **Note**
>
> MacOS Catalina를 사용하는 경우, Firefox 웹 브라우저를 사용합니다. Chrome은 지원되지 않습니다.

14. 접속이 비공개 또는 안전하지 않다는 메시지가 표시되면 보안 경고를 수락하고 호스트 페이지로 진행합니다.

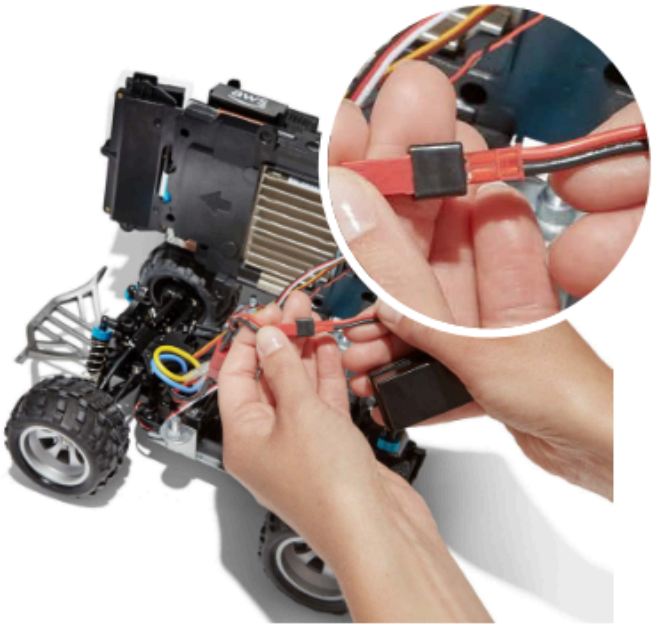
이제 AWS DeepRacer이 Wi-Fi에 연결되었습니다.

AWS DeepRacer 주행 모듈 배터리를 충전하는 방법

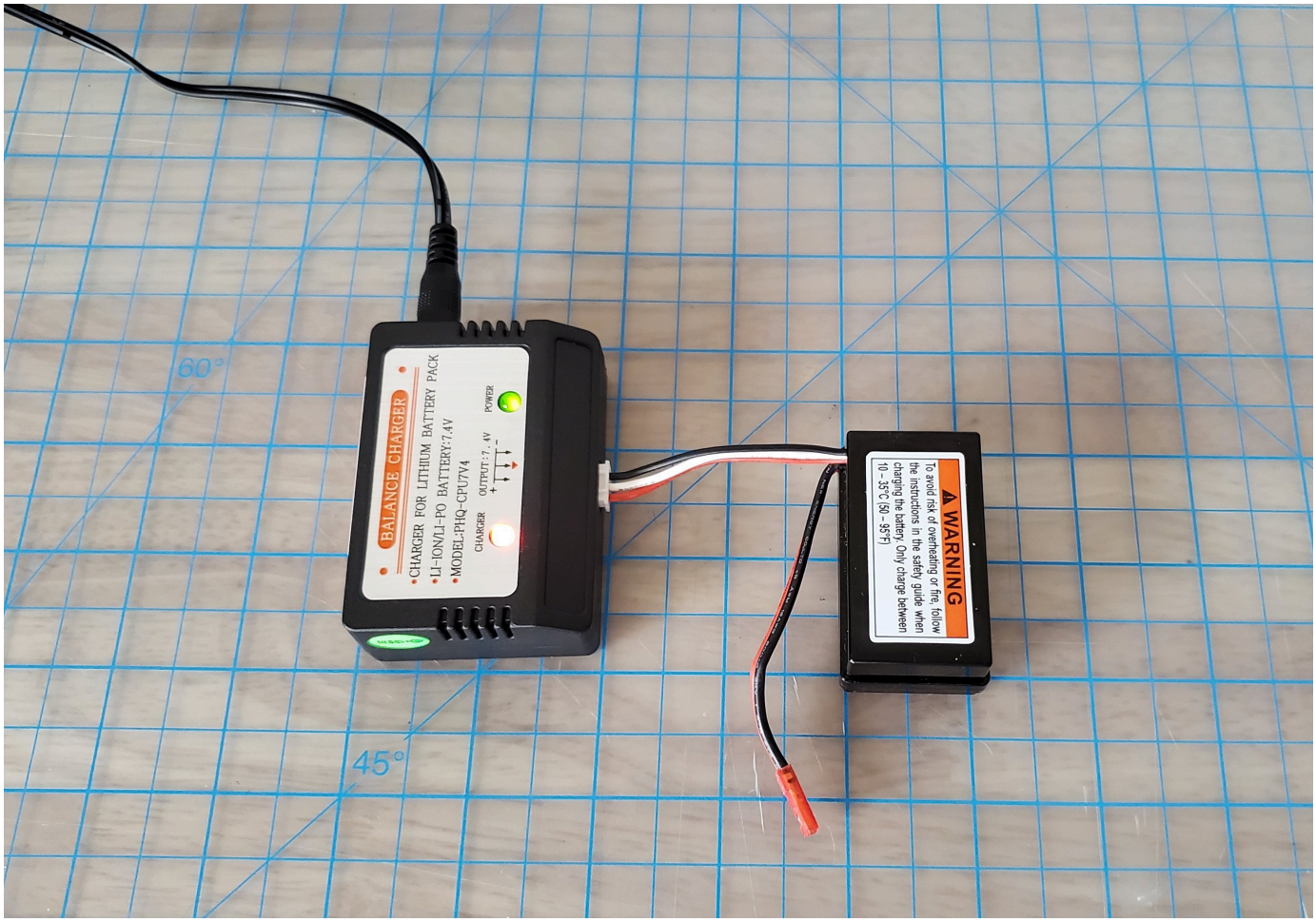
AWS DeepRacer 드라이브 모듈 배터리에는 흰색과 빨간색의 두 색상이 적용된 JST 커넥터가 있는 두 세트의 케이블이 있습니다. 검은색, 빨간색, 흰색 케이블 끝에 있는 흰색 3핀 커넥터는 차량 모듈 배터리를 배터리 충전기에 연결합니다. 검은색과 빨간색 케이블 끝에 있는 빨간색 2핀 커넥터는 배터리를 차량 드라이브 트레인에 연결합니다.

AWS DeepRacer 주행 모듈 배터리는 아래 단계에 따라 충전합니다.

1. 차량에 연결된 드라이브 모듈 배터리에 접근하려면 드라이브 트레인에 연결되는 전선이 느슨해지지 않도록 주의하면서 컴퓨팅 모듈을 들어 올리십시오.



2. 선택 사항으로 차량과 드라이브 모듈 배터리를 분리하려면 검은색 및 빨간색 드라이브 트레인 커넥터에서 빨간색 2핀 배터리 커넥터를 분리하고 벨크로 스트랩을 풀면 됩니다.
3. 배터리의 흰색 3핀 커넥터를 충전기 포트에 연결하여 배터리를 배터리 충전기에 연결합니다.



빨간색 표시등 + 녹색 표시등 = 완전히 충전되지 않음

4. 배터리 충전기의 전원 코드를 콘센트에 꽂습니다. 녹색 표시등만 켜지면 배터리가 완전히 충전된 것입니다.
5. 충전 어댑터에서 충전된 차량 배터리의 흰색 3핀 커넥터를 분리합니다. 배터리를 분리하여 충전한 경우(선택 사항) 빨간색 2핀 커넥터를 차량 드라이브 트레인 커넥터에 다시 연결하고 벨크로 스트랩으로 배터리를 차량에 고정하십시오.
6. 스위치를 "ON" 위치로 밀어서 차량 드라이브 트레인을 켭니다. 신호음이 짧게 두 번 울리는지 들어보아 차량이 성공적으로 충전되었는지 확인합니다. 신호음이 두 번 들리지 않으면 차량 배터리를 잠금 해제해 보십시오.

이제 AWS DeepRacer 드라이브 모듈 배터리를 사용할 준비가 되었습니다.

AWS DeepRacer 컴퓨팅 모듈 배터리를 충전하는 방법

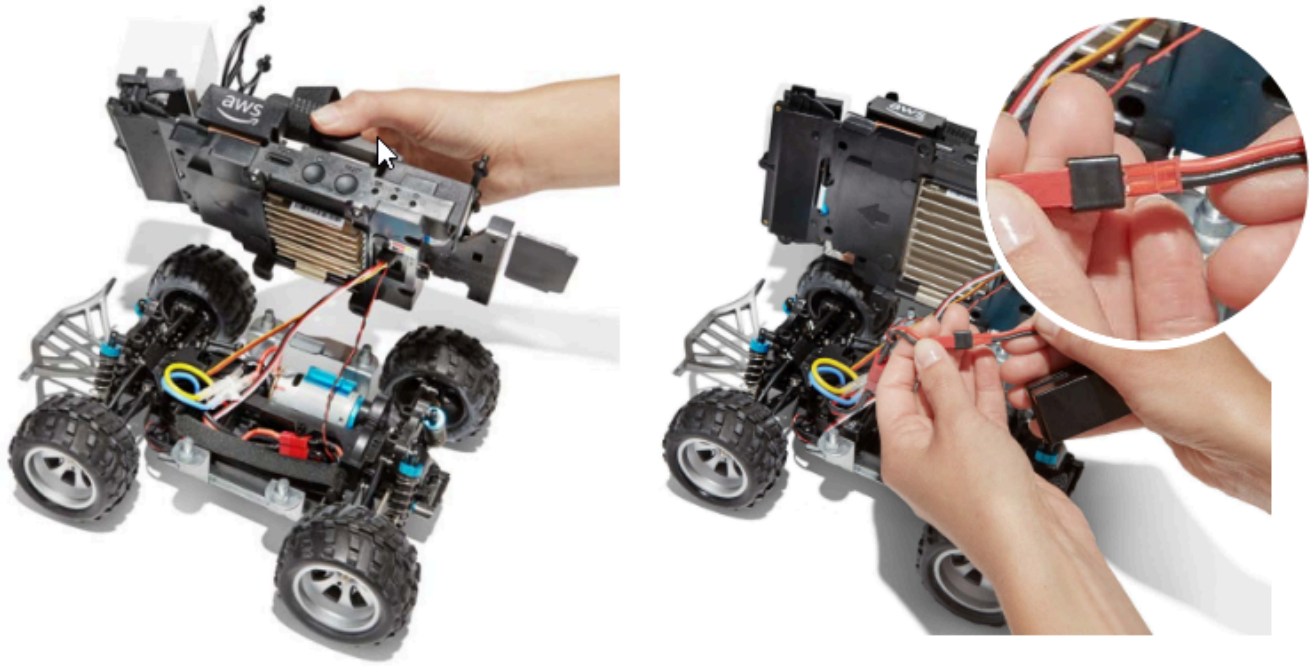
AWS DeepRacer 컴퓨팅 모듈 배터리는 아래 단계에 따라 충전합니다.

1. 경우에 따라 차량에서 컴퓨팅 모듈 배터리를 분리합니다.
2. 컴퓨팅 모듈 충전기를 컴퓨팅 모듈 배터리에 연결합니다.
3. 컴퓨팅 배터리 충전기의 전원 코드를 콘센트에 꽂습니다.

배터리가 충전되었지만 AWS DeepRacer 차량이 움직이지 않음

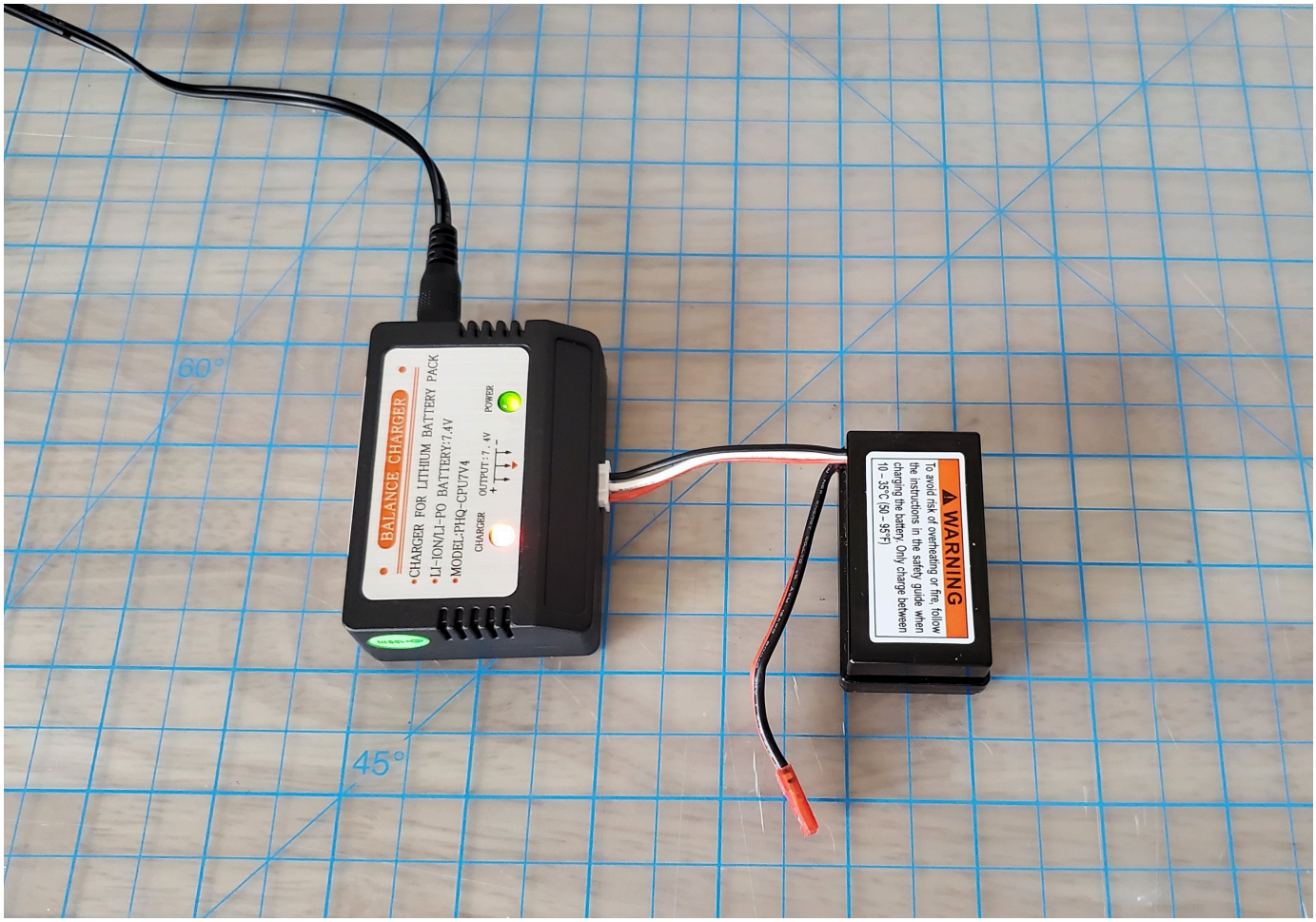
AWS DeepRacer 콘솔이 설정되고, 컴퓨팅 배터리가 충전되고, Wi-Fi가 연결되어 있지만 차량이 여전히 움직이지 않는 경우 다음 단계를 따르십시오.

1. 컴퓨팅 모듈을 들어 올립니다. 이 모듈을 드라이브 트레인에 연결하는 전선이 느슨해지지 않도록 주의하십시오. 아래에 있는 차량 배터리가 올바르게 연결되어 있는지 확인하십시오. 빨간색 2핀 커넥터는 검은색 및 빨간색 드라이브 트레인 커넥터에 연결되어 있습니다.



2. 스위치를 “ON” 위치로 눌러 차량 드라이브 트레인을 켭니다. 신호음이 짧게 두 번 울리는지 들어보아 차량이 충전되었는지 확인합니다. 차량 전원이 성공적으로 켜지면 4단계로 건너뜁니다.

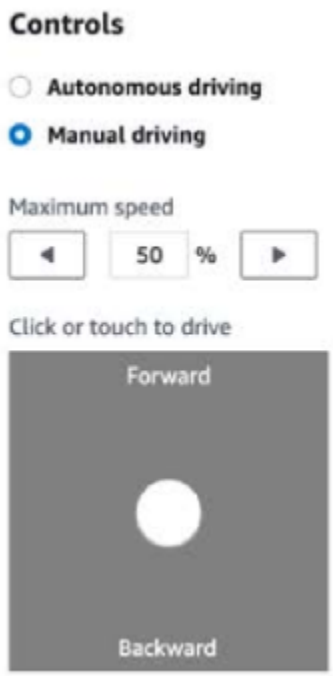
3. 차량 배터리를 켤 때 신호음이 두 번 울리지 않으면 배터리가 완전히 충전되었는지 확인합니다. 차량 배터리의 흰색 커넥터 케이블을 충전 어댑터에 꽂습니다. 충전 어댑터는 빨간색 및 녹색 LED 표시등으로 컴퓨팅 모듈의 어댑터와 구분할 수 있습니다. 어댑터를 충전 케이블에 연결하고 전원 콘센트에 꽂습니다. 차량 배터리 충전 어댑터의 빨간색 및 녹색 표시등이 모두 켜지면 배터리를 충전해야 함을 나타냅니다.



빨간색 표시등 + 녹색 표시등 = 완전히 충전되지 않음

녹색 표시등만 켜지면 배터리가 완전히 충전된 것이므로 사용할 수 있습니다. 충전 어댑터에서 자동차 배터리의 흰색 커넥터를 분리하고 빨간색 커넥터를 차량에 다시 연결합니다. 배터리를 분리하여 충전한 경우(옵션) 벨크로 스트랩으로 다시 드라이브 트레인에 고정하십시오. 스위치를 "ON" 위치로 밀어서 차량 드라이브 트레인을 켭니다. 그래도 신호음이 두 번 들리지 않으면 차량 배터리를 잠금 해제해 보십시오.

4. 차량을 Wi-Fi에 연결하고 브라우저에서 AWS DeepRacer 콘솔을 엽니다. 터치 조이스틱으로 차량을 수동으로 운전하여 움직일 수 있는지 확인합니다.



주의 사항: 차량 배터리에서 최대한의 주행 거리를 얻으려면 AWS DeepRacer를 사용하지 않을 때는 차량 드라이브 트레인을 끄거나 배터리를 분리하십시오.

차량이 여전히 움직이지 않으면 AWSDeepRacer-Help@amazon.com으로 문의하십시오.

AWS DeepRacer 차량 배터리 잠금 문제를 해결합니다.

> **Important**
>
> 이 배터리는 DeepRacer 자동차에만 사용할 수 있습니다. 이 배터리는 화재, 폭발 또는 기타 안전 문제를 방지하기 위해 올바르게 취급되어야 합니다. 모든 지침을 따르고 AWS DeepRacer 디바이스 안전 가이드에 포함된 모든 경고에 유의하십시오.

AWS DeepRacer 디바이스 약관, 보증 및 고지

- AWS DeepRacer 디바이스 사용 약관
- AWS DeepRacer 디바이스 1년 제한 보증
- AWS DeepRacer 디바이스 안전 가이드

주제

- AWS DeepRacer 차량 배터리 잠금을 방지하는 방법
- 잠긴 후 AWS DeepRacer 차량 배터리를 잠금 해제하는 방법

AWS DeepRacer 차량 배터리 잠금을 방지하는 방법

AWS DeepRacer 차량 배터리 잠금을 방지하는 방법을 알아봅니다.

배터리 상태를 유지하기 위해 AWS DeepRacer 차량 배터리가 잠금 상태로 전환됩니다. 이 경우 배터리가 부분적으로 충전되어 있더라도 배터리는 차량에 전원을 공급하지 않습니다. 자동차 배터리가 잠금 상태로 전환되지 않도록 하려면 다음을 수행하십시오.

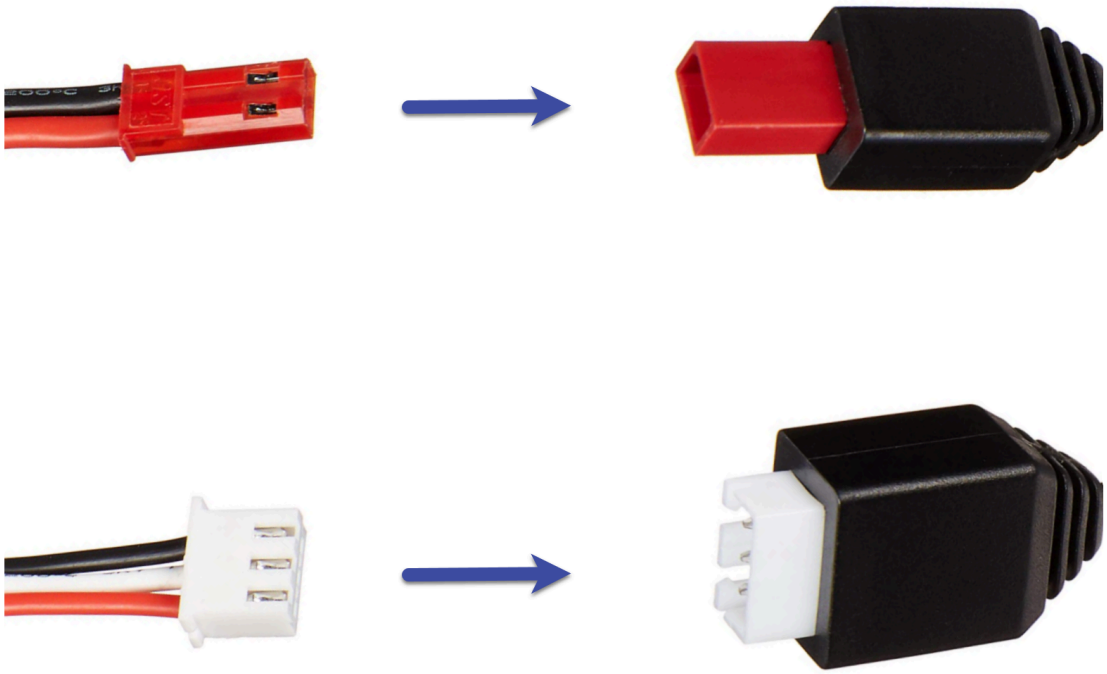
- AWS DeepRacer 사용을 완료하면 배터리 충전을 유지하기 위해 차량을 끕니다.
- 디바이스 콘솔에 차량 배터리의 전력 수준이 낮다는 경고가 표시되면 최대한 빨리 충전하십시오.
- AWS DeepRacer를 잠시 사용하지 않을 것이라고 생각되면 배터리를 차량에서 분리하고 완전히 충전하십시오. 차량 배터리를 보호하고 잠기지 않도록 최소 일 년에 한 번 충전하는 것이 좋습니다.

> **Note**
>
> 모든 리튬 폴리머(LiPo) 배터리는 사용하지 않을 때에도 시간이 지남에 따라 천천히 방전됩니다.

잠긴 후 AWS DeepRacer 차량 배터리를 잠금 해제하는 방법

잠긴 후 AWS DeepRacer 배터리를 잠금 해제하려면 잠금 해제 케이블을 사용하십시오.

1. 배터리 커넥터를 해당 색상의 케이블 커넥터(빨간색은 빨간색, 흰색은 흰색)에 삽입합니다.



2. 케이블에서 배터리를 분리합니다.

3. AWS DeepRacer 차량 배터리를 즉시 사용할 수 있습니다. 빨간색 2핀 커넥터를 차량 드라이브 트레인 커넥터에 다시 연결하고 벨크로 스트랩으로 배터리를 차량에 고정합니다.
4. 스위치를 "ON" 위치로 밀어서 차량 드라이브 트레인을 켭니다. 신호음이 짧게 두 번 울리는지 들어보아 배터리가 성공적으로 잠금 해제되었는지 확인합니다.

LiDAR 센서를 설치할 때 Dell 배터리 커넥터 케이블을 감싸는 방법

각도를 조절할 수 있는 매우 긴 Dell USB-C 커넥터 케이블을 사용하여 AWS DeepRacer 차량에 연결된 LiDAR 센서 위에 Evo 쉘을 장착하려면 특정 케이블 래핑 기술이 필요합니다.

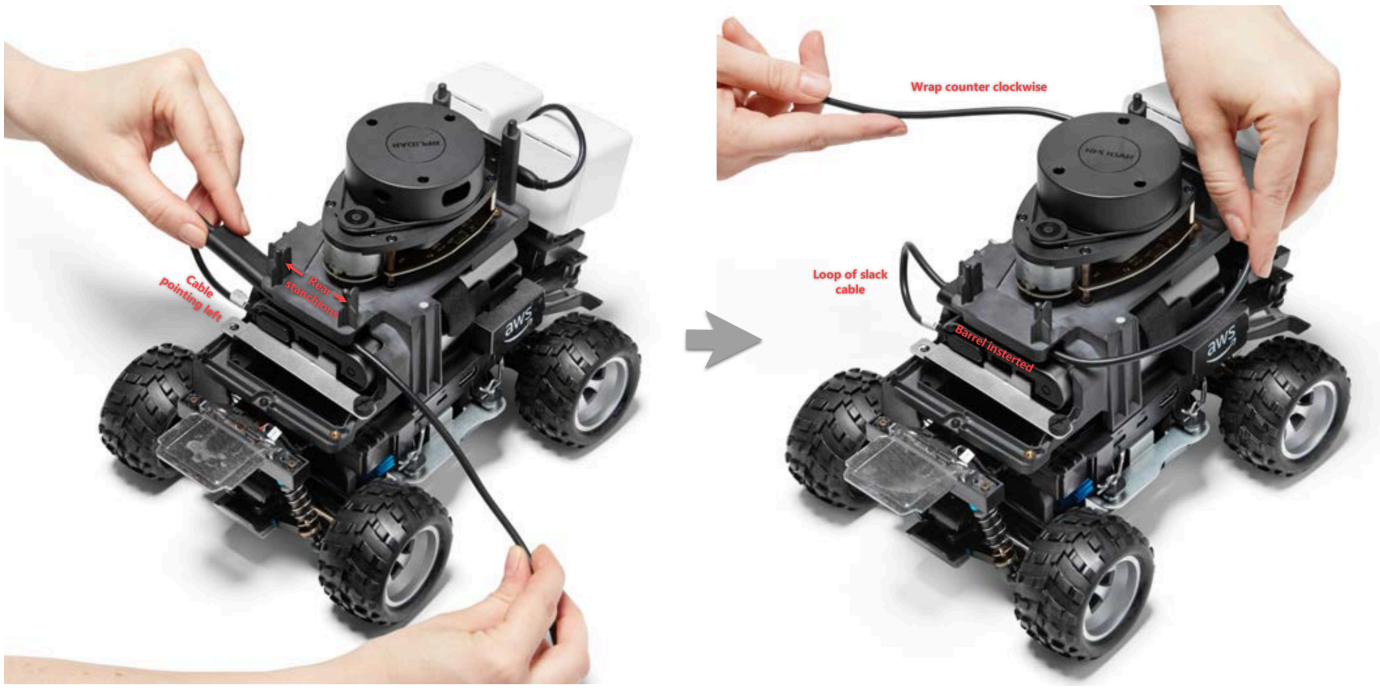
이 프로세스에 대한 비디오를 보려면 YouTube에서 AWS DeepRacer: LiDAR 센서 설치 및 Dell 컴퓨팅 배터리 커넥터 케이블 래핑을 참조하십시오. 동영상은 AWS DeepRacer 차량에 LiDAR 센서를 설치하는 것으로 시작합니다. Dell 배터리 래핑 기법은 00:01:27 초에 시작됩니다.

> **Note**
>
> Dell 컴퓨팅 배터리 커넥터 케이블에는 배럴, 표준 USB-C 엔드, 앵글 USB-C 엔드가 있습니다.

Evo 쉘을 장착할 수 있도록 Dell 배터리 케이블을 LiDAR 센서에 감는 방법

1. AWS DeepRacer 차량 후면을 향하고, 커넥터 케이블이 왼쪽을 향하도록 하여 컴퓨팅 배터리 커넥터 케이블의 앵글 끝을 컴퓨팅 배터리 USB-C 포트에 꽂습니다.

2. 차량을 약간 왼쪽으로 돌려 LiDAR 홀더와 컴퓨팅 배터리 사이의 후면 지지대 바로 아래에 있는 스페이스를 찾아 케이블을 통과시킵니다. 배럴이 이 스페이스에 삽입되면 케이블을 당기는 것을 중지하십시오. USB-C 포트 왼쪽에 느슨한 케이블 고리가 있어야 합니다.



3. AWS DeepRacer 차량의 후면을 향하고, 케이블을 시계 반대 방향으로 LiDAR 센서 베이스에 감고 케이블 클립을 사용하여 케이블이 꼭 맞도록 고정합니다.
4. 차량을 약간 오른쪽으로 돌리고 케이블의 표준 USB-C 끝을 USB-C 포트에 꽂습니다.

5. Evo 쉘을 AWS DeepRacer 차량에 장착하고 핀으로 고정하여 잘 장착되었는지 테스트합니다. 쉘이 올바르게 장착되면 쉘의 컷아웃을 통해 LiDAR 센서를 완전히 볼 수 있으며 지지대 상단의 핀 구멍에 접근할 수 있습니다. 쉘을 제거하고 필요에 따라 케이블을 조정하십시오.

LiDAR 센서가 연결되었습니다. 이제 차량의 전원을 켜고, 운전하고, 실험할 준비가 되었습니다.

차량의 Wi-Fi 연결을 유지하는 방법

다음 문제 해결 안내서에서는 차량의 연결을 유지하기 위한 팁을 제공합니다.

차량의 Wi-Fi LED 표시기가 파란색으로 깜박인 다음 2초 동안 빨간색으로 바뀐 뒤 마지막으로 꺼진 경우 Wi-Fi 연결 문제를 해결하는 방법

다음을 점검해 Wi-Fi 연결 설정이 올바른지 확인합니다.

- USB 드라이브에 wifi-creds.txt 파일이 하나만 있고 디스크 파티션이 하나만 있는지 확인합니다. wifi-creds.txt 파일이 여러 개 있는 경우 전부 발견된 순서대로 처리되어 예기치 않은 동작으로 이어질 수 있습니다.

- wifi-creds.txt 파일에서 Wi-Fi 네트워크의 SSID 및 암호가 올바르게 지정되어 있는지 확인합니다. 예를 들어, 이 파일은 다음과 같은 모양입니다.

```
################################################################################
#                                   AWS DeepRacer                              #
# File name: wifi-creds.txt                                                    #
#                                                                              #
# ...                                                                          #
################################################################################

# Provide your SSID and password below
ssid: ' MyHomeWi-Fi'
password: myWiFiPassword
```

- wifi-creds.txt 파일에서 `ssid`과 `password`의 필드 이름이 모두 소문자인지 확인하십시오.
- 각 필드 이름과 값은 콜론 하나(:)로 구분되어야 합니다. 예: `ssid : ' MyHomeWi-Fi'`
- 공백이 포함된 필드 값은 작은 따옴표 한 쌍으로 묶여 있는지 확인합니다. Mac에서 TextEdit 또는 기타 텍스트 편집기에서는 작은 따옴표가 '...' 형식이 아니라 '...' 형식으로 표시됩니다. 필드 값에 공백이 없는 경우 해당 값에는 작은 따옴표가 없을 수 있습니다.

차량의 Wi-Fi 또는 전원 LED 표시기가 파란색으로 깜박이면 무슨 의미입니까?

USB 드라이브에 wifi-creds.txt 파일이 포함되어 있으면 차량이 이 파일에 지정된 Wi-Fi 네트워크에 연결을 시도하는 동안 Wi-Fi LED 표시기가 파란색으로 깜박입니다.

USB 드라이브에 `models` 디렉터리가 있으면 차량이 이 디렉터리 내로 모델 파일을 로드하려고 시도하는 동안 전원 LED가 파란색으로 깜박입니다.

USB 드라이브에 wifi-creds.txt 파일 및 `models` 디렉터리가 둘 다 있으면 차량이 두 가지 작업을 순차적으로 처리하는데, Wi-Fi 연결을 먼저 시도한 다음 모델을 로드합니다.

Wi-Fi 연결 시도에 실패하면 Wi-Fi LED가 2초 동안 빨간색으로 변할 수 있습니다.

어떻게 하면 호스트 이름을 사용하여 차량의 디바이스 콘솔에 연결할 수 있습니까?

호스트 이름을 사용하여 차량의 디바이스 콘솔에 연결하는 경우 브라우저에 `https://hostname.local`이라고 입력합니다. 여기서 *hostname* 값(`AMSS-1234` 형식)은 AWS DeepRacer 차량의 맨 아래에 인쇄되어 있습니다.

IP 주소를 사용하여 차량의 디바이스 콘솔에 연결하는 방법

(USB 드라이브에 있는) device-status.txt 파일에 표시된 IP 주소를 사용하여 디바이스 콘솔에 연결하려면 다음 조건을 충족해야 합니다.

- 랩톱 또는 모바일 디바이스가 AWS DeepRacer 차량과 동일한 네트워크에 있는지 확인합니다.
- VPN에 연결되어 있는지 확인하고, 연결되어 있는 경우 먼저 연결을 끊습니다.
- 다른 Wi-Fi로 시도하십시오. 예를 들어, 휴대폰에서 개인 핫스팟을 켭니다.

AWS DeepRacer 디바이스의 MAC 주소를 가져오는 방법

아래 지침에 따라 AWS DeepRacer 디바이스의 MAC 주소를 가져옵니다.

1. 이때 AWS DeepRacer 디바이스는 Wi-Fi 네트워크에만 연결되어 있어야 합니다.
2. AWS DeepRacer 디바이스를 모니터에 연결합니다. HDMI-HDMI, HDMI-DVI 또는 유사한 케이블이 필요합니다. 케이블의 한쪽 단부를 차량의 섀시에 있는 HDMI 포트에 삽입하고, 나머지 한쪽 단부를 모니터에서 지원되는 디스플레이 포트에 연결합니다.
3. 컴퓨팅 모듈이 부팅되면 디바이스의 컴퓨팅 모듈에 있는 USB 포트를 사용해 USB 키보드를 AWS DeepRacer에 연결합니다.
4. 사용자 이름 입력 필드에 `deepracer`을 입력합니다.
5. 디바이스 SSH 암호를 암호 입력 필드에 입력합니다.

   디바이스에 처음 로그인하는 경우 암호 입력 필드에 `deepracer`를 입력합니다. 다음 단계로 이동하기 전에 필요하다면 암호를 재설정합니다. 그러면 이후 로그인해서는 새로운 암호를 사용하게 됩니다. 새로운 암호로 재설정할 때는 보안을 이유로 복잡하거나 강력한 암호 문구를 사용해야 합니다.
6. 로그인한 후 터미널 창을 엽니다.

   터미널 애플리케이션에 검색 버튼을 사용할 수 있습니다.

7. 다음 Ubuntu 셸 명령을 터미널 창에 입력합니다.

```
ifconfig | grep HWaddr
```

명령을 실행하면 다음과 비슷하게 출력됩니다.

```
mlan0    Link encap:Ethernet   HWaddr   01:2a:34:b5:c6:de
```

16진수가 디바이스의 MAC 주소입니다.

AWS DeepRacer 디바이스 콘솔의 기본 암호를 복구하는 방법

AWS DeepRacer 디바이스 콘솔의 기본 암호를 복구할 때는 기본 암호를 가져오거나 재설정해야 합니다. 기본 암호는 다음 그림과 같이 디바이스 아래쪽에 인쇄되어 있습니다.

다음 절차에 따라 Ubuntu 터미널 창에서 AWS DeepRacer 디바이스 웹 서버의 암호를 복구하십시오.

1. AWS DeepRacer 디바이스를 모니터에 연결합니다. HDMI-HDMI, HDMI-DVI 또는 유사한 케이블이 필요합니다. 케이블의 한쪽 단부를 차량의 섀시에 있는 HDMI 포트에 삽입하고, 나머지 한쪽 단부를 모니터에서 지원되는 디스플레이 포트에 연결합니다.

2. 컴퓨팅 모듈이 부팅되면 디바이스의 컴퓨팅 모듈에 있는 USB 포트를 사용해 USB 키보드를 AWS DeepRacer에 연결합니다.
3. 사용자 이름에 `deepracer`를 입력합니다.
4. Password(암호)에 디바이스 SSH 암호를 입력합니다.

   디바이스에 처음 로그인하는 경우 암호에 `deepracer`를 입력합니다. 다음 단계로 이동하기 전에 필요하다면 암호를 재설정합니다. 그러면 이후 로그인해서는 새로운 암호를 사용하게 됩니다. 새로운 암호로 재설정할 때는 보안을 이유로 복잡하거나 강력한 암호 문구를 사용해야 합니다.
5. 로그인한 후 터미널 창을 엽니다.

   검색 버튼을 사용하여 터미널 창 애플리케이션을 찾을 수 있습니다.
6. 디바이스 콘솔의 기본 암호를 가져오려면 터미널 창에 다음 명령을 입력합니다.

   ```
   $cat /sys/class/dmi/id/chassis_asset_tag
   ```

   명령을 실행한 결과로 기본 암호가 출력됩니다.

7. 디바이스 콘솔 암호를 기본 암호로 재설정하려면 터미널 창에서 다음 Python 스크립트를 실행합니다.

   ```
   sudo python /opt/aws/deepracer/nginx/reset_default_password.py
   ```

AWS DeepRacer 디바이스를 수동으로 업데이트하는 방법

AWS DeepRacer 서비스의 최근 변경 사항으로 인해 AWS re:Invent 2018에 배포된 디바이스와 같은 특정 레거시 디바이스가 자동으로 업데이트될 수 없습니다. 아래 단계를 따라 이러한 디바이스를 수동으로 업데이트합니다.

AWS DeepRacer 디바이스를 수동으로 업데이트하려면

1. AWS DeepRacer 디바이스 스크립트 수동 업데이트를 컴퓨터에 다운로드하고 압축을 풉니다.

   이 스크립트에 대한 압축되지 않은 파일의 기본 이름은 `deepracer-device-manual-update.sh`입니다. 이 주제에서는 이 기본 스크립트 파일 이름을 사용한다고 가정하겠습니다.
2. 다운로드하고 압축되지 않은 스크립트 파일(`deepracer-device-manual-update.sh`)을 컴퓨터에서 USB 드라이브로 복사합니다.

3. HDMI-HDMI 케이블을 사용하여 모니터, USB 키보드, USB 마우스에 디바이스를 연결합니다.
4. 디바이스의 전원을 켜고, 디바이스가 부팅된 후 OS에 로그인합니다.

   디바이스에 처음 로그인하는 경우, 새로운 OS 암호를 설정해야 합니다.
5. 디바이스에 USB 드라이브를 연결하고 디바이스의 폴더(예: ~/Desktop)로 스크립트 파일을 복사합니다.
6. 디바이스의 터미널에서 다음 명령을 입력하여 스트립트 파일 폴더로 이동하고 스크립트 파일에 실행 권한을 추가합니다.

   ```
   cd ~/Desktop
   chmod +x deepracer-device-manual-update.sh
   ```
7. 다음 쉘 명령을 입력하여 스크립트를 실행합니다.

   ```
   sudo -H ./deepracer-device-manual-update.sh
   ```
8. 디바이스 업데이트가 끝나면 컴퓨터 또는 모바일 디바이스에서 웹 브라우저를 열고 디바이스 IP 주소로 이동합니다(예: 홈 네트워크의 192.168.1.11 또는 사무실 네트워크의 10.56.101.13).

   다바이스가 Wi-Fi 네트워크에 연결되어 있고 VPN을 통해 터널링하지 않고 동일한 네트워크에 있는 브라우저를 사용합니다.
9. 디바이스 콘솔에서 디바이스 콘솔에 로그인할 때 사용하는 암호를 입력합니다. 업데이트 화면이 나타나길 기다립니다. 추가 업데이트에 대한 메시지가 표시되면 해당 지침을 따릅니다.

일반적인 AWS DeepRacer 운영 문제를 진단하고 해결하는 방법

AWS DeepRacer 차량을 이용한 강화 학습을 살펴보는 도중에 디바이스가 작동을 멈추는 경우가 있습니다. 다음 문제 해결 주제는 문제를 진단하고 해결하는 데 도움이 됩니다.

주제

- 디바이스 콘솔의 비디오 플레이어에 내 차량 카메라의 비디오 스트림이 표시되지 않는 이유는 무엇입니까?
- 내 AWS DeepRacer 차량이 움직이지 않는 이유는 무엇입니까?
- 최신 디바이스 업데이트가 표시되지 않는 이유는 무엇입니까? 최신 업데이트를 받으려면 어떻게 해야 합니까?

- 내 AWS DeepRacer 차량이 내 Wi-Fi 네트워크에 연결되지 않는 이유는 무엇입니까?
- AWS DeepRacer 디바이스 콘솔 페이지의 로딩 시간이 오래 걸리는 이유는 무엇입니까?
- AWS DeepRacer 차량에 배포할 때 모델이 제대로 작동하지 않는 이유는 무엇입니까?

디바이스 콘솔의 비디오 플레이어에 내 차량 카메라의 비디오 스트림이 표시되지 않는 이유는 무엇입니까?

AWS DeepRacer 디바이스 콘솔에 로그인했는데, 디바이스 제어의 비디오 플레이어에 AWS DeepRacer 차량에 장착된 카메라의 실시간 비디오 스트림이 표시되지 않습니다. 다음과 같은 이유로 이 문제가 발생할 수 있습니다.

- 카메라와 USB 포트의 연결이 끊겼을 수 있습니다. 차량에서 카메라 모듈을 뽑았다가 USB 포트에 다시 꽂고, 디바이스의 전원을 껐다가 다시 켜서 재시작합니다.
- 카메라 결함일 수 있습니다. 다른 AWS DeepRacer 차량에서 정상 작동하는 것이 확인된 카메라를 사용하여 이러한 경우에 해당하는지 테스트해 봅니다.

내 AWS DeepRacer 차량이 움직이지 않는 이유는 무엇입니까?

AWS DeepRacer 차량의 전원을 켰지만 차량이 움직이지 않습니다. 다음과 같은 이유로 이 문제가 발생할 수 있습니다.

- 차량의 파워뱅크를 켜지 않았거나, 파워뱅크가 차량에 연결되어 있지 않습니다. 파워뱅크의 USB C 포트와 차량 섀시의 USB-C 포트를 제공된 USB C-USB C 케이블로 연결합니다. 파워뱅크의 충전 수준을 알리는 LED 표시등이 켜졌는지 확인합니다. 불이 들어오지 않았으면 파워뱅크의 전원 버튼을 누른 다음 차량 섀시의 전원 버튼을 눌러 디바이스를 부팅합니다. 미등이 켜지면 디바이스가 부팅된 것입니다.
- 파워뱅크가 켜져 있고 차량도 부팅했는데 수동 주행 모드와 자동 주행 모드 중 어느 모드에서도 차량이 움직이지 않으면 차량 섀시 아래의 배터리가 충전된 상태이고 켜져 있는지 확인하십시오. 충전되어 있지 않으면 차량 배터리를 다시 충전하고 배터리가 완전히 충전된 후 전원을 켭니다.
- 차량 배터리 케이블 커넥터가 디바이스 주행 모듈의 전원 케이블 커넥터에 완전히 꽂혀 있지 않습니다. 케이블 커넥터가 단단히 연결되어 있는지 확인하십시오.
- 배터리 케이블에 결함이 있습니다. 가능하면 정상 작동하는 다른 차량에서 이 배터리를 테스트하여 이러한 경우에 해당하는지 확인합니다.
- 차량 배터리의 전원 스위치를 켜지 않았습니다. 전원 스위치를 켜고, 신호음이 두 번 울린 후에 다시 길게 울리는지 확인합니다.

최신 디바이스 업데이트가 표시되지 않는 이유는 무엇입니까? 최신 업데이트를 받으려면 어떻게 해야 합니까?

AWS DeepRacer 차량의 소프트웨어가 오래된 버전인 이유는 무엇입니까?

- 최근 들어 디바이스에서 자동 업데이트가 수행되지 않았습니다. 수동 업데이트를 수행해야 할 수 있습니다.
- 차량이 인터넷에 연결되어 있지 않습니다. 차량이 인터넷 액세스가 가능한 Wi-Fi 또는 이더넷 네트워크에 연결되어 있는지 확인하십시오.

내 AWS DeepRacer 차량이 내 Wi-Fi 네트워크에 연결되지 않는 이유는 무엇입니까?

차량 OS에서 네트워크 상태를 확인했는데 AWS DeepRacer 차량이 어떤 Wi-Fi 네트워크에도 연결되지 않은 것으로 나옵니다. 이 현상은 다음과 같은 문제 때문일 수 있습니다.

- AWS DeepRacer 차량의 Wi-Fi를 구성하지 않았습니다. 이 설정 지침에 따라 차량의 Wi-Fi 네트워크를 설정하십시오.
- 차량이 활성 네트워크 신호 범위를 벗어났습니다. 선택한 Wi-Fi 네트워크 범위 안에서 차량을 작동해야 합니다.
- 차량의 사전 구성된 Wi-Fi 네트워크가 사용 가능한 Wi-Fi 네트워크와 맞지 않습니다. 설정 지침에 따라 활성 CAPTCHA가 필요 없는 Wi-Fi 네트워크를 재설정하십시오.

AWS DeepRacer 디바이스 콘솔 페이지의 로딩 시간이 오래 걸리는 이유는 무엇입니까?

AWS DeepRacer 차량의 디바이스 콘솔을 열려고 하는데 디바이스 콘솔 페이지의 로딩 시간이 오래 걸리는 것 같습니다.

- 차량이 멈췄거나 꺼져 있습니다. 미등을 켤 때 차량 전원이 켜져 있는지 확인하십시오.
- 네트워크의 DHCP 서버에 의해 차량의 IP 주소가 바뀌었을 가능성이 높습니다. 차량의 새 IP 주소를 찾아보려면 이 설정 지침에 따라 컴퓨터와 차량을 USB-US 케이블로 연결하여 디바이스 콘솔에 로그인하십시오. 설정에 새 IP 주소가 나와 있습니다. 아니면 네트워크에 연결된 디바이스 목록을 검사하여 새 IP 주소를 확인할 수도 있습니다. 본인이 네트워크 관리자가 아니라면 관리자에게 확인을 요청하십시오.

AWS DeepRacer 차량에 배포할 때 모델이 제대로 작동하지 않는 이유는 무엇입니까?

모델을 훈련하고 모델의 아티팩트를 내 AWS DeepRacer 차량에 배포한 후 차량이 예상대로 작동하지 않는 경우가 있습니다. 무엇이 문제였습니까?

일반적으로 물리적 AWS DeepRacer 차량으로 전송하도록 교육된 모델을 최적화하는 것은 까다로운 학습 프로세스입니다. 종종 시행 착오를 반복해야 합니다. 모범 사례에 대한 일반적인 지침은 실제 환경에 대한 AWS DeepRacer 모델 훈련 최적화 단원을 참조하십시오.

다음은 AWS DeepRacer 차량의 모델 성능에 영향을 미칠 수 있는 몇 가지 일반적인 요소입니다.

- 모델이 교육에 수렴되지 않았습니다. 모델을 복제하여 교육을 계속하거나 장기간 모델을 다시 교육하십시오. 에이전트가 시뮬레이션에서 랩을 계속 완료하는지, 즉 훈련이 끝날 때까지 100% 프로세스를 완료하는지 확인합니다.
- 모델이 과도하게 훈련되었습니다(즉, 과도하게 적합함). 훈련 데이터에는 너무 적합하지만 알 수 없는 상황으로 일반화되지 않습니다. 보다 유연하거나 적응력이 뛰어난 보상 함수를 사용하여 모델을 다시 훈련하거나 행동 공간의 세부 수준을 높이십시오. 또한 훈련된 모델을 다른 트랙에서 평가하여 모델이 잘 일반화되었는지 확인해야 합니다.
- AWS DeepRacer 차량이 제대로 보정되지 않았을 수 있습니다. 이것이 사실인지 테스트하려면 수동 주행으로 전환한 후 차량이 예상대로 주행하는지 확인하십시오. 예상대로 주행하지 않는 경우 차량을 보정하십시오.
- 요구 사항을 충족하지 않는 트랙에서 자율적으로 차량을 운행하고 있습니다. 트랙 요구 사항은 AWS DeepRacer를 위한 물리적 트랙 빌드 단원을 참조하십시오.
- 물리적 트랙 근처에 물체가 너무 많아서 트랙이 시뮬레이션 환경과 상당히 다릅니다. 트랙 주변을 정리하여 물리적 트랙을 시뮬레이션 환경과 가능한 한 유사하게 만듭니다.
- 트랙 표면이나 가까이 있는 물체에서 반사되어 눈부심이 생겨서 카메라를 혼란시킬 수 있습니다. 조명을 조정하고 매끄러운 표면의 콘크리트 바닥 또는 다른 반짝이는 재료로 트랙을 만들지 마십시오.

AWS DeepRacer 개발자 안내서용 문서 이력

다음 표에서는 AWS DeepRacer의 최신 릴리스가 발표된 이후 이 설명서에서 변경된 중요 사항에 대해 설명합니다.

변경 사항	설명	날짜
2023년 AWS DeepRacer 리그 업데이트	AWS DeepRacer 리그를 참조하는 여러 주제를 업데이트했습니다. 2023년 AWS DeepRacer 리그 시즌에 대한 자세한 내용은 이용 약관을 참조하십시오.	2023년 3월 1일
SageMaker AI 노트북을 사용하여 AWS DeepRacer 모델 훈련 및 평가 주제가 일시적으로 제거됨 SageMaker	SageMaker AI 노트북을 사용하여 AWS DeepRacer 모델 훈련 및 평가 주제가에서 제거되었습니다모델 훈련 및 평가. 현재 AWS DeepRacer AWS 에서 SageMaker AI 노트북을 사용하는 절차가 업데이트되고 있습니다.	2022년 11월 1일
다중 사용자 기능에 대한 IAM 관리형 정책 업데이트	새로운 관리형 정책 `AWSDeepRacerAccountAdminAccess` 및 `AWSDeepRacerDefaultMultiUserAccess` 가 추가되어 다중 사용자 모드를 사용하여 하나의 AWS DeepRacer 계정으로 여러 참가자를 후원할 수 있습니다. the section called “AWSDeepRacerAccountAdminAccess” 섹션을 참조하십시오.	2021년 10월 26일

다중 사용자 기능 업데이트	AWS DeepRacer는 이제 한 AWS 계정이 여러 참가자가 레이스하고 훈련할 수 있도록 지원하는 다중 사용자 기능을 지원합니다. 자세한 내용은 다중 사용자 모드 단원을 참조하십시오.	2021년 10월 26일
다중 차량 레이스 및 장애물 회피 업데이트	AWS DeepRacer는 다중 차량 레이스 및 장애물 회피를 가능하게 하는 새로운 센서 유형인 스테레오 카메라와 LIDAR를 지원합니다. 자세한 내용은 the section called “레이스 유형 및 이를 가능하게 하는 센서 이해” 섹션을 참조하십시오.	2019년 12월 2일
커뮤니티 레이스 업데이트	이제 AWS DeepRacer를 사용하여 AWS DeepRacer 사용자가 커뮤니티 레이스라고 하는 자체 레이싱 이벤트를 구성할 수 있으며, 개인 리더보드는 초대된 사용자에게만 공개됩니다. 자세한 내용은 레이스 참가 섹션을 참조하십시오.	2019년 12월 2일
정식 출시를 위한 업데이트	딥 러닝 모델을 더욱 견고하게 훈련하고 평가할 수 있는 메소드가 AWS DeepRacer에 추가되었습니다. 사용자 인터페이스가 업데이트되면서 설명이 추가되었습니다. 나만의 물리적 트랙을 빌드할 수 있는 옵션과 정밀한 데이터가 추가되었습니다. 문제 해결 정보가 제공됩니다.	2019년 4월 29일

최초 릴리스의 AWS DeepRacer 개발자 안내서	AWS DeepRacer 사용자가 강화 학습에 대해 배우는 동시에 AWS DeepRacer 콘솔, AWS DeepRacer 시뮬레이터 및 AWS DeepRacer 모형 차량을 사용해 자율 레이싱에 강화 학습을 적용하는 방법을 살펴보도록 돕기 위해 마련된 문서의 최초 릴리스입니다.	2018년 11월 28일

AWS 용어집

최신 AWS 용어는 AWS 용어집 참조의 AWS 용어집을 참조하세요.

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.