

DeepRacing Local Training Set-Up

공통적으로 해야할 것

1. Linux 설치

https://www.youtube.com/watch?v=S_J5yi-JVpY&t=274s

2. Docker 설치

<https://blog.cosmosfarm.com/archives/248/우분투-18-04-도커-docker-설치-방법>

3. Docker-Compose 설치

```
curl -L https://github.com/docker/compose/releases/download/1.24.1/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
```

아마 위에 그대로 옮기면 실행이 안될 수도 있다. 이때는 리눅스의 shell을 root계정으로 바꿔서 실행해주면 된다. root계정의 password 설정 방법 및 root 계정으로의 전환 방법은 구글에 많이 있다.

그리고 이제, docker-compose가 잘 설치되었는지 확인하기 위해 밑과 같은 코드를 shell에 입력해주면 된다.

`docker-compose --version`

만약 잘 설치가 되었다면 현재 리눅스에 설치되어 있는 docker-compose의 버전이 뜰 것이다.

그런데, 여기서 root 계정을 해제하고 다시 일반계정에서 docker-compose를 쓰려고 하면 'ERROR Couldn't connect to Docker daemon'라는 메시지가 뜨면서 문제가 생긴다. 이는 docker의 사용권한을 root계정만 가지고 있기 때문이다. 일반계정도 사용권한을 주면 해결되는 가벼운 문제이다. 해결 방법은 아래의 링크를 타고 들어가서 그대로 따라하면 된다.

<https://codechacha.com/ko/fix-couldnt-connect-to-docker-daemon/>

1. deepracer-for-dummies (2019.ver)

<https://github.com/aws-deepracer-community/deepracer-for-dummies>

<https://www.youtube.com/watch?v=CFNcKmtVRSI> (explaining through Youtube Video)

2. deepracer-local (2020.ver)

<https://github.com/mattcamp/deepracer-local/>

3. AWS Training (Cost)

<https://aws.amazon.com/ko/deepracer/>

* If you have gpu(Nvidia), I recommend 1.

* If you are a rich guy, I recommend 3.

* I am not both of them.. So I used 2.

각 제목 밑의 첨부해놓은 링크를 타고 들어가면 상세하게 설명되어있다.

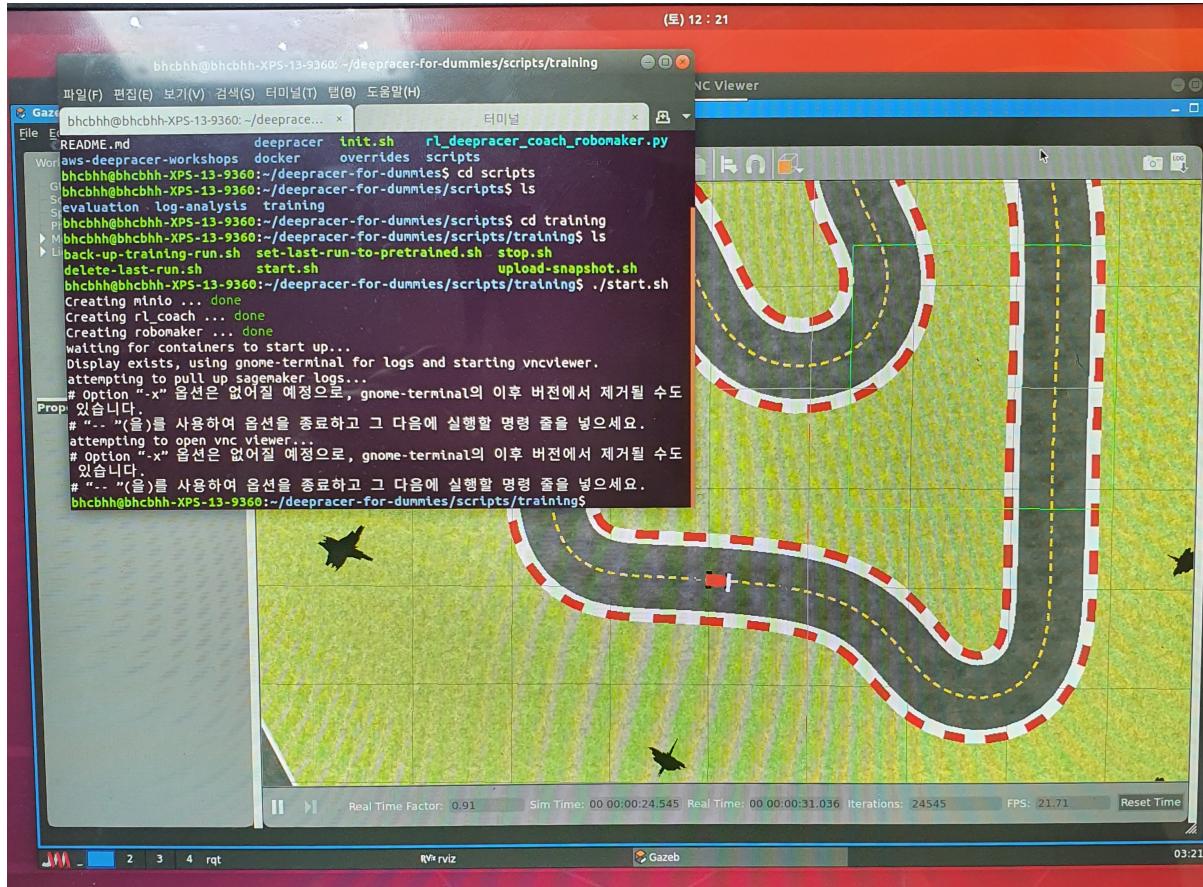
(기본적으로 맨 위에 설명해놓은 3가지는 이미 되어있는 상태에서 설치과정을 따라하는 것을 추천)

여기서부터는, 각 과정을 할 때마다, 가장 빈번하게 만나는 오류에 대해서 설명하겠다.

유의해야 점은 local-training 모두 사용자가 gpu가 있다고 생각하고 있다. 즉, nvidia gpu가 장착되지 않은 사용자들에게는 설명이 배려되어 있지 않다. gpu를 사용하지 않고 cpu로 위의 시뮬레이션을 돌릴 경우 시뮬레이션 영상 상에서 멈춤현상이 심하게 나타날 것이라고 경고되어 있다. (그렇다고 사용하면 안된다는 것은 아니다 라는 말도 같이 있다.)

1. deepracer-for-dummies (2019.ver)

나의 경우는 1.deepracer-for-dummies를 했을 당시에는 차량이 움직이지 않기도 하고, 차량이 track을 벗어났음에도 위치가 reset되지 않았다.



이에, deepracer 관련 커뮤니티 사람들에게 조언을 구하였고 2. deepracer-local 을 사용하라는 말을 듣고 현재 나의 리눅스에는 2. deepracer-local이 다운로드 되어 있다.

5 replies



Lyndon Leggate 24 hours ago

You don't need to use GPU for local training, but it will run slower without it. It should still run though.



Lyndon Leggate 24 hours ago

This repo is the latest/best one to use:

<https://github.com/mattcamp/deepracer-local>



GitHub

[mattcamp/deepracer-local](#)

Contribute to mattcamp/deepracer-local development by creating an account on GitHub.



Matt Camp 24 hours ago

Honestly at this stage I think you are better off switching to one of the newer repos such as my one at <https://github.com/mattcamp/deepracer-local/> as the deepracer-for-dummies setup is the one from last year.

But the answer to your issue with the car not resetting can probably be found by looking at the logs for the robomaker and sagemaker containers. If you don't have a GPU then the policy training can take quite a while and during this time on the old setup the car will often just drive off the track or randomly spin around



GitHub

[mattcamp/deepracer-local](#)

Contribute to mattcamp/deepracer-local development by creating an account on GitHub.



Matt Camp 24 hours ago

`docker ps` to show running containers, and then `docker logs -f <container id>` to tail the logs

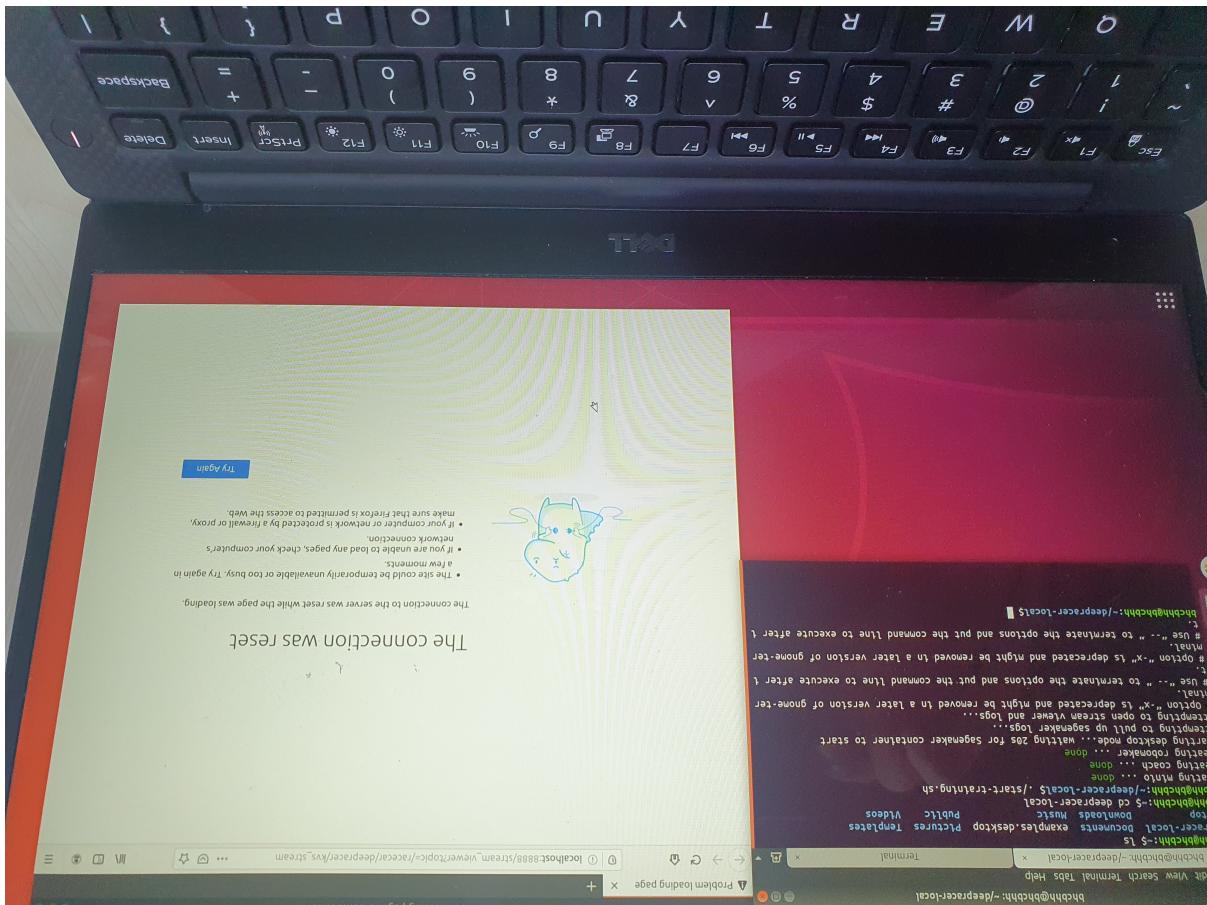
2. deepracer-local (2020.ver)

deepracer-local에 있는 과정을 모두 따라한 후, 폴더를 deepracer-local로 이동.

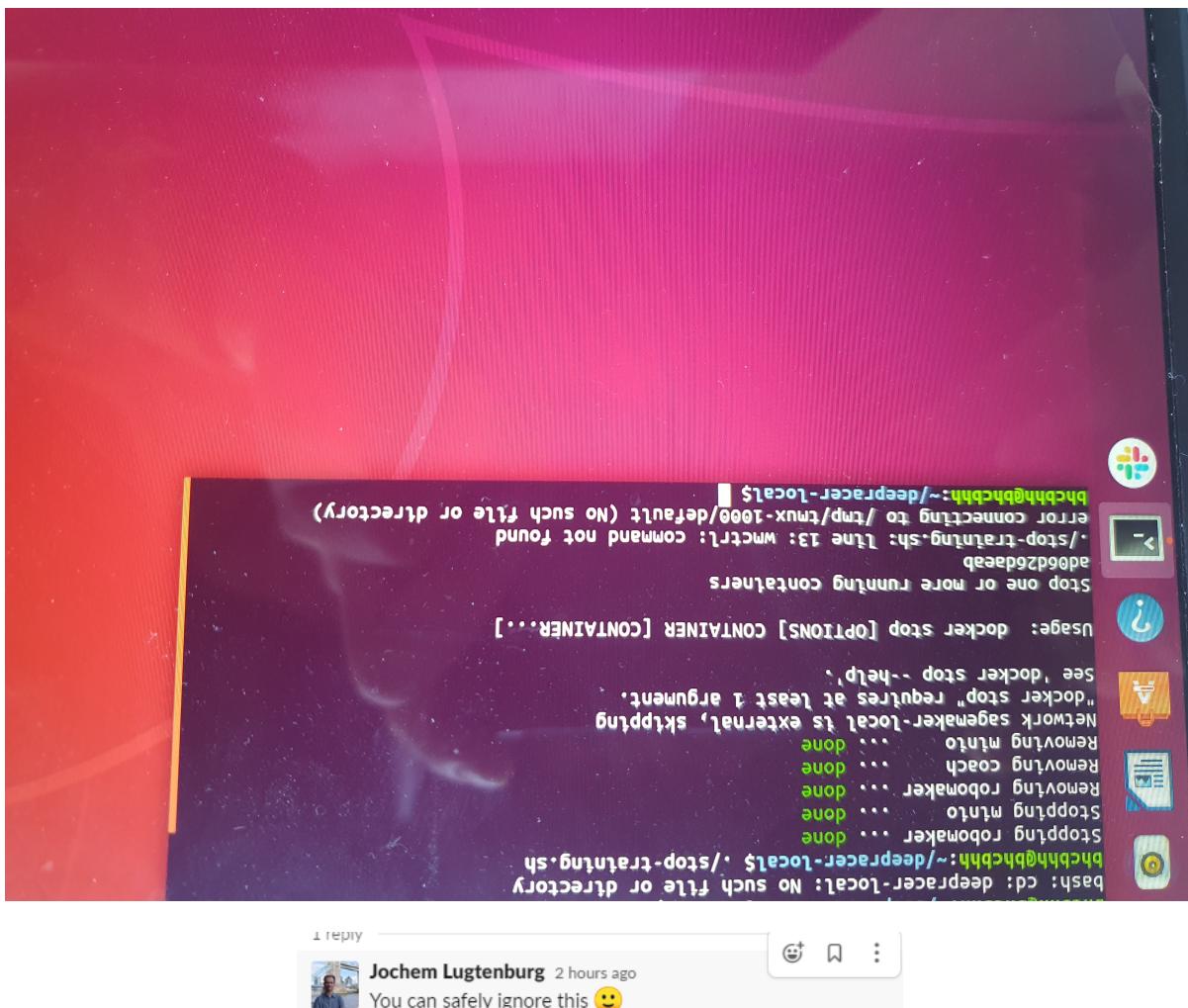
리눅스 shell에 `./start-training.sh`를 입력하면 밑의 사진과 같은 오류가 나온다.

이에, deepracer-local 폴더의 `./delete_last_run.sh`를 실행 한 후에 다시 `./start-training.sh`를 실행하였더니 다시 동작되었다.

(참고로, 위와 같은 페이지 에러가 나타날 경우 새로고침을 하면 화면이 뜨는 것을 확인할 수 있다.)



밑의 error는 training을 종료한 후, (./end-training.sh) 뜬 오류다. 이 또한 커뮤니티 사람들에게 물어봤더니 아래와 같은 대답이 돌아왔다.



그리고, 여기서 cpu를 사용하는 경우에 주의 해야할 부분이 있다. default값으로 gpu가 설정되어 있기 때문에 몇 가지를 바꾸어줘야 한다. 이 또한 링크를 타고 들어가는 github에 친절하게 설명되어 있다.

- tweak any other settings you want in `config.env`
 - Modify `ENABLE_GPU_TRAINING` for SageMaker runtime: `true` (nvidia runtime) or `false` (CPU runtime). Default is GPU.
 - If you do not have an nvidia GPU then you will also need to change the tag of the robomaker image inside `docker-compose.yml`
 - Set `ENABLE_LOCAL_DESKTOP` to `true` if you have a local X-windows install (desktop machine) and want to automatically start the stream viewer and tail sagemaker logs.
 - Install tmux (`sudo apt install tmux` on Ubuntu Linux) if you want robomaker + sagemaker logs automatically tailed in your terminal session.

위의 사진에 나타나는 지침들은 되도록이면 전부 따라하는 것을 권장한다. (cpu를 사용할 경우)

밀의 동영상은 deepracer-local을 성공했을 때의 영상이다. 밀의 터미널을 보게 되면 차량이 장애물에 부딪혔을 때 episode가 증가되어 log들이 밀려 올라가는 것을 확인할 수 있다.

<https://www.youtube.com/watch?v=wIAVQDbLBs4&feature=youtu.be>

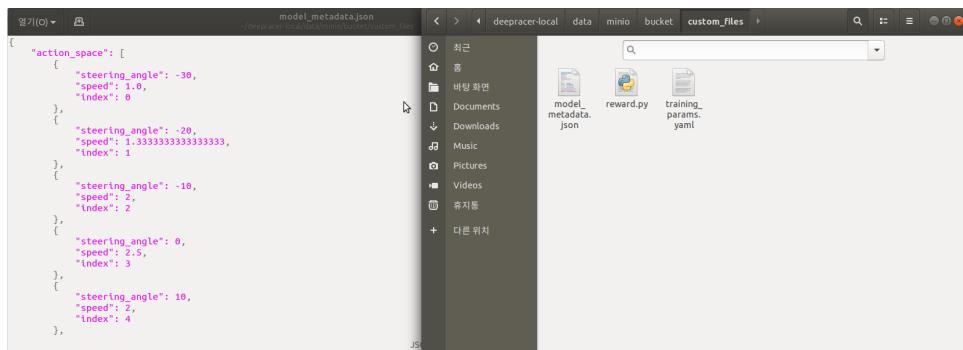
세부적인 사용방법은 아직 2020ver이기 때문에 커뮤니티 사람들 사이에서도 매뉴얼 같은 것이 없다고 한다. 앞으로 지속적인 사용을 통해서 알아가는 것이 필요할 듯 하다. 그래도 관련하여 추천받은 링크를 아래에 달아놓겠다.

<https://wiki.deepracing.io>

https://wiki.deepracing.io/Customise_Local_Training

* 주요 폴더 경로

1) Action Space



2) Reward Function

```
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_distance = params['objects_distance']
    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
```

3) HyperParameter

```

rl_deepracer_coach_robomaker.py
~/deepracer-local/src/rcoach_2020_v2
sagemaker_session=sagemaker.Session(),
#bypass sagemaker SDK validation of the role
role="arn:aws:iam::aaa:root",
train_instance_type=instance_type,
train_instance_count=1,
output_path=s3_output_path,
base_job_name=job_name,
image_name=image_name,
train_max_run=job_duration_in_seconds, # Maximum run time
hyperparameters={"s3_bucket": s3_bucket,
                 "s3_prefix": s3_prefix,
                 "aws_region": aws_region,
                 "model_metadata_s3_key": "s3://{}-{}/customer/rcoach_2020_v2/{}".format(s3_bucket),
                 "RLCOACH_PRESET": RLCOACH_PRESET,
                 "batch_size": 64,
                 "beta_entropy": 0.01,
                 "discount_factor": 0.999,
                 "epsilon_greedy_value": 0.05,
                 "epsilon_steps": 10000,
                 "exploration_type": "categorical",
                 "loss_type": "mean squared error",
                 "lr": 0.0003,
                 "num_episodes_between_training": 20,
                 "num_epochs": 10,
                 "stack_size": 1,
                 "term_cond_avg_score": 1000000.0,
                 "term_cond_max_episodes": 100000
}

```

* Local - training이 어떤식으로 진행되는지

1) 초반부 (약 2시간 짜리)

<https://youtu.be/SuwRP12Tlws>

2) 10시간이 지나고 난 후

<https://youtu.be/fF5jza0xRGs>

* 20.04 Local Training 관련해서 추가본

우선 nvidia driver와 cuda를 설치해야 한다. 현재 리눅스는 20.04를 사용하고 있고 이는 굉장히 잘 설명되어 있는 링크가 있기에 첨부해놓았다.

1. nvidia : <https://linuxconfig.org/how-to-install-the-nvidia-drivers-on-ubuntu-20-04-focal-fossa-linux>
2. cuda : <https://linuxconfig.org/how-to-install-cuda-on-ubuntu-20-04-focal-fossa-linux>

잘 설치되었는지 확인하려면 아래의 명령어를 shell에 입력해주면 된다.

nvidia-smi

```

lab_kaai@LabkaAI:~$ nvidia-smi
Thu Jun 11 19:32:12 2020
+-----+
| NVIDIA-SMI 440.64      Driver Version: 440.64      CUDA Version: 10.2 |
+-----+
| GPU  Name      Persistence-M | Bus-Id     Disp.A  | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap| Memory-Usage | GPU-Util  Compute M. |
|=====                         ======                 ======  ====== |
|  0  TITAN X (Pascal)      Off | 00000000:88:00.0 Off |          N/A |
| 28%   43C    P0    56W / 250W |      0MiB / 12196MiB |     0%       Default |
+-----+
|  1  TITAN X (Pascal)      Off | 00000000:89:00.0 Off |          N/A |
| 28%   42C    P0    57W / 250W |      0MiB / 12196MiB |     0%       Default |
+-----+
| Processes:                               GPU Memory |
| GPU  PID  Type  Process name        Usage        |
| =====|=====|=====|=====|
| No running processes found
+-----+
lab_kaai@LabkaAI:~$
```

위에 첨부한 사진들은 나의 컴퓨터에서 local-training을 설치한 것이다. 하지만, 현재 참고에 있던 컴퓨터 하나를 DeepRacing용으로 할당받았기에 이를 사용하기 위해서 해당 컴퓨터에도 local-training을 설치해야 했다. 기존의 위의 설치 과정과 차이가 있다면 gpu를 사용해야 한다는 점이다. 자세한 사항들은 첨부한 github에 들어가면 확인할 수 있다. 다만 자세히 나와있지 않은 nvidia-docker2의 설치 방법에 대해서는 설명이 필요할 듯 하여 작성한다.

`sudo install nvidia-docker2`

를 입력하면 설치가 알아서 될 것이다. 그리고 이를 잘 설치하였는 지 확인하기 위해서는 /etc/docker 경로에 daemon.json이 잘 설치되었는지 확인하면 된다. 만약, 제대로 설치가 되었다면 해당 파일이 폴더안에 존재할 것이다. gpu 드라이버의 경우 자칫 잘못하면 드라이버가 충돌하여 리눅스를 재설치해야하는 상황까지 갈 수 있으므로 신중하게 진행하는 것을 추천한다.

만약, 위의 명령어 입력으로도 잘 설치가 되지 않았다면 아래의 github을 방문하여 순서를 따라하면 될 것이다.

관련링크 : <https://github.com/NVIDIA/nvidia-docker>

원격 컴퓨터에서 로컬 트레이닝을 성공 영상

https://youtu.be/gTA_CtvpLG4

* 로컬 트레이닝 해석 참고

```
bhcbhh@bhcbhh:~/deepracer-local$ ./deep_racer_local.py --track=racetrack --agent=main_level --log_level=INFO
[286, Training iteration=1
Training> Name=main_level/agent, Worker=0, Episode=23, Total reward=56.02, Steps
=298, Training iteration=1
Training> Name=main_level/agent, Worker=0, Episode=24, Total reward=107.03, Step
s=320, Training iteration=1
Training> Name=main_level/agent, Worker=0, Episode=25, Total reward=132.03, Step
s=347, Training iteration=1
Training> Name=main_level/agent, Worker=0, Episode=26, Total reward=57.02, Steps
=359, Training iteration=1
Training> Name=main_level/agent, Worker=0, Episode=27, Total reward=57.02, Steps
=371, Training iteration=1
Training> Name=main_level/agent, Worker=0, Episode=28, Total reward=52.01, Steps
=382, Training iteration=1
Training> Name=main_level/agent, Worker=0, Episode=29, Total reward=65.02, Steps
=396, Training iteration=1
Training> Name=main_level/agent, Worker=0, Episode=30, Total reward=61.02, Steps
=409, Training iteration=1
Training> Name=main_level/agent, Worker=0, Episode=31, Total reward=95.02, Steps
=429, Training iteration=1
Training> Name=main_level/agent, Worker=0, Episode=32, Total reward=73.02, Steps
=444, Training iteration=1
Training> Name=main_level/agent, Worker=0, Episode=33, Total reward=58.01, Steps
=456, Training iteration=1
```

step : DeepRacer 찍은 사진 1개 (이런 사진을 1초에 굉장히 많이 찍는다.)

episdoe : step이 모인 것, 차량이 track을 나갈 때 까지

iteration : 차량이 1바퀴를 완주했을 때 (해당 숫자는 local Training에서 Evaluation이 끝나면 올라간다.)