AWS DeepRacer-Basic

陳昀(Ingrid)

GitHub LinkedIn

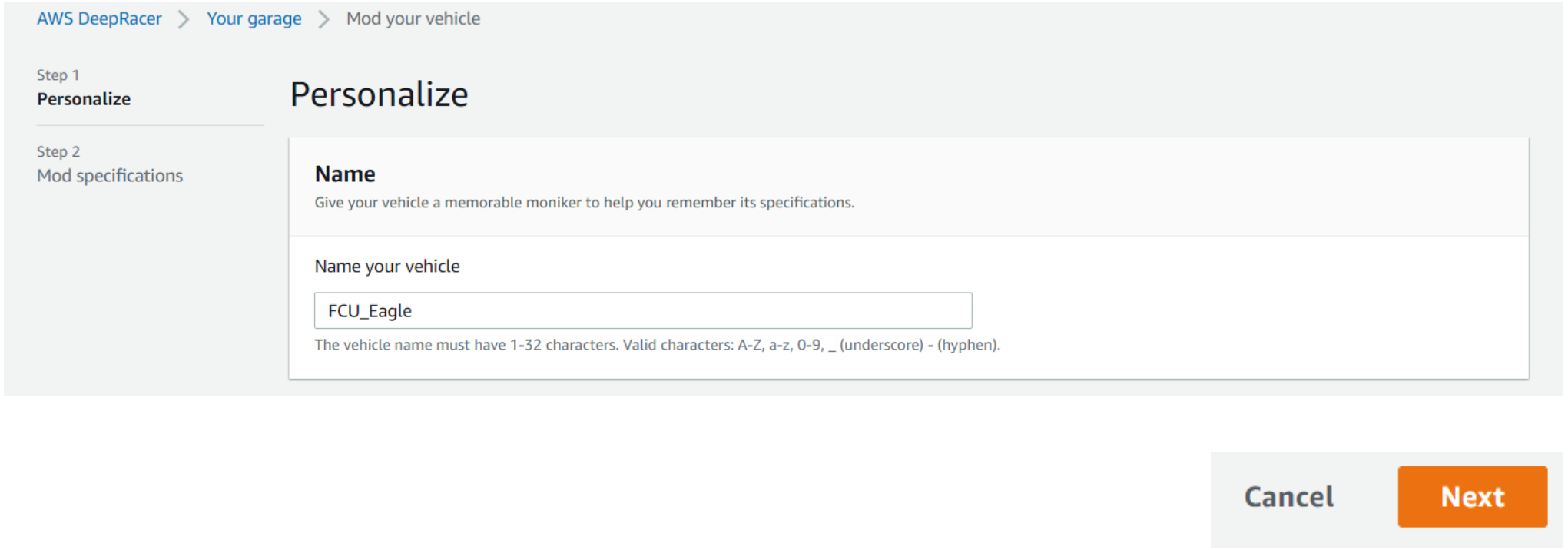
aws certified Cloud Practitioner

aws certified Solutions Architect Associate

Lab

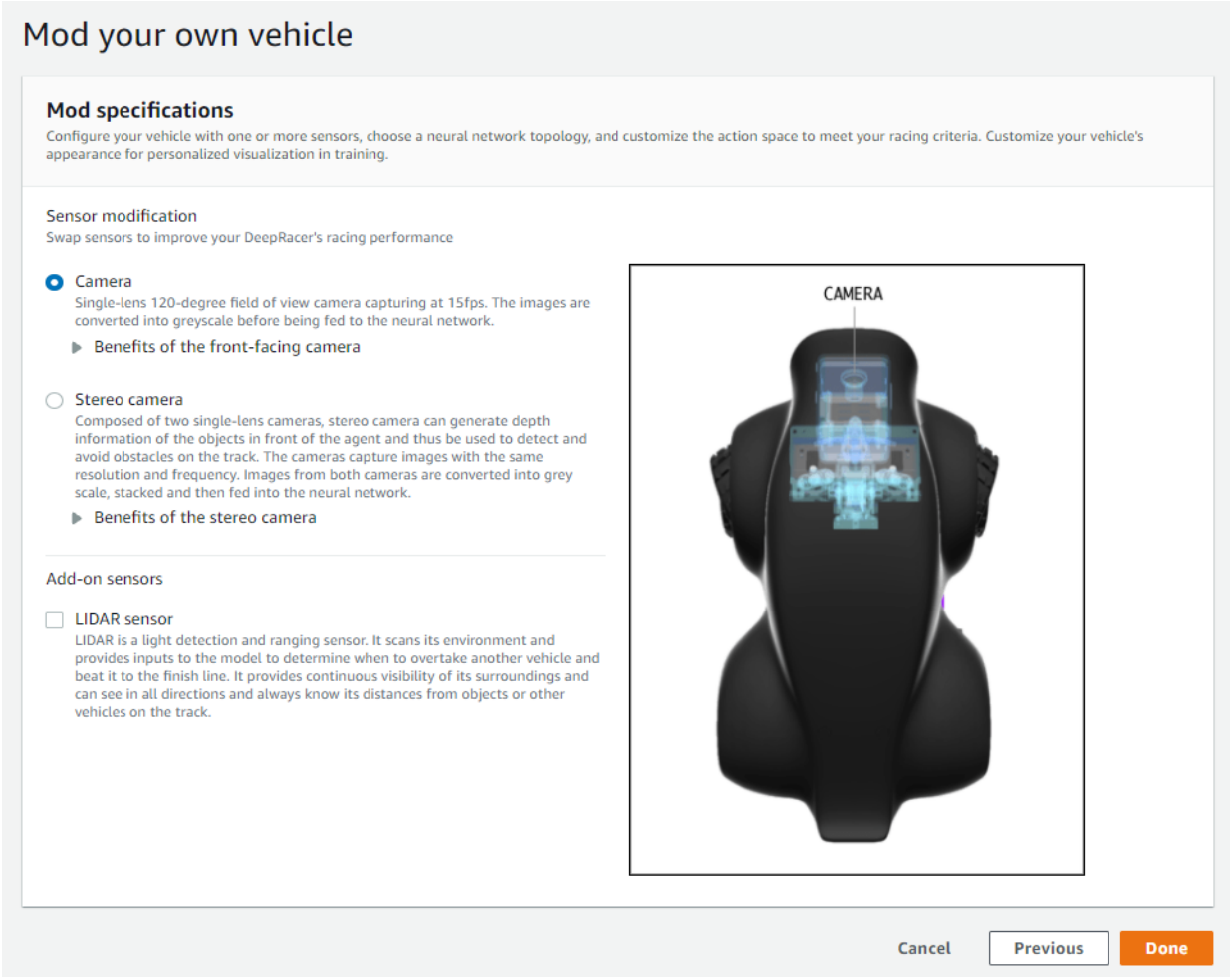
- **Find the lab content here: Link**
- **Create a car**
- **Start training your time-trial model**
- **Submit your model to the Virtual Race**

Your garage- Personalize

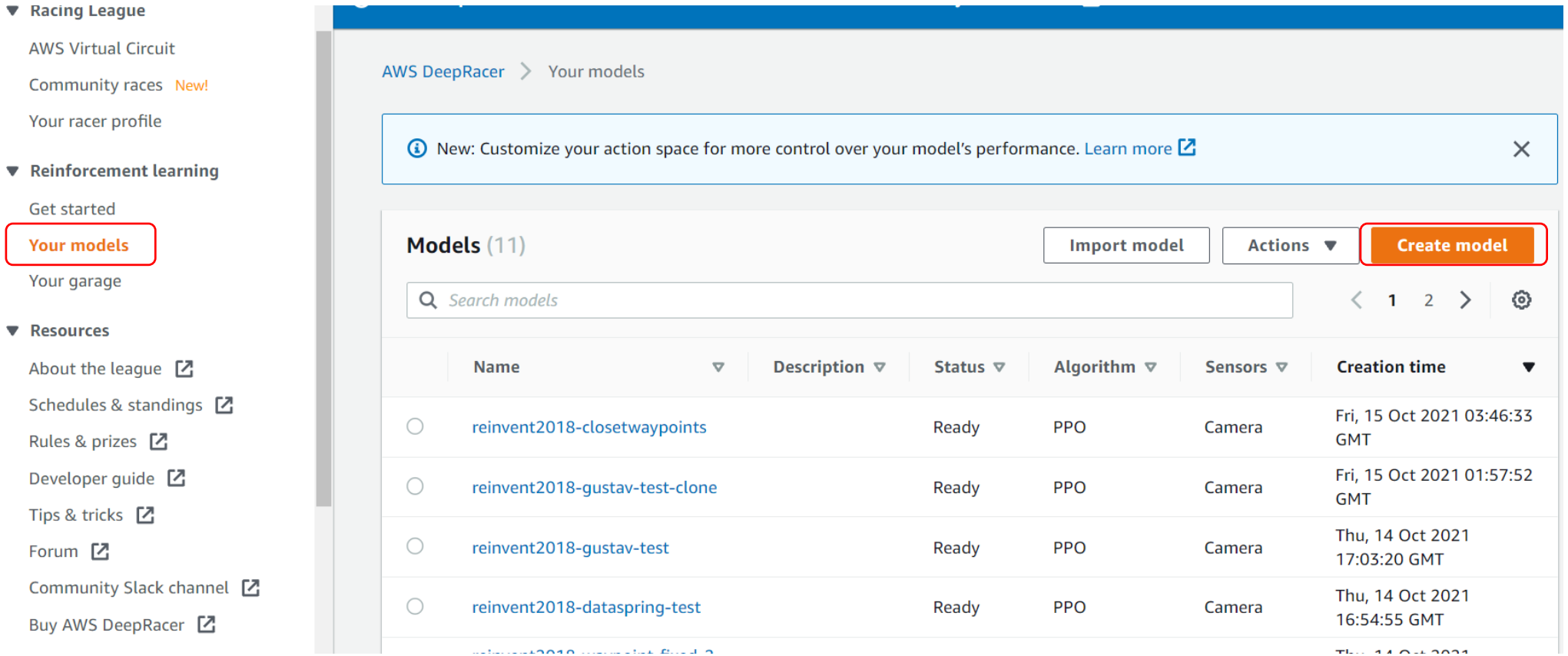


Your garage- Mod your own vehicle

- **Camera**
- **Stereo camera**
- **LIDAR sensor**



Your models- Create model



Your models- Specify the model name and environment

- Model name

-

Training details

Model name

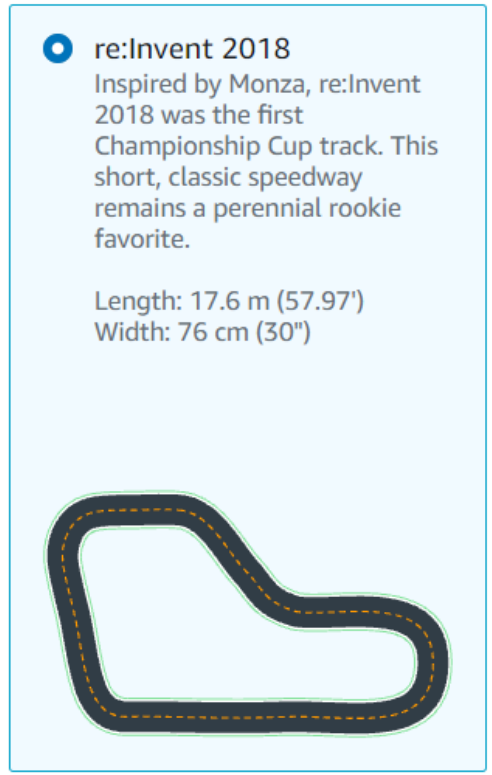
TopModel-re:InventTrack

The model name must be unique and can have up to 64 characters. Valid characters are a-z, A-Z, 0-9, and - (hyphen). No spaces or underscores.

Training job description - optional

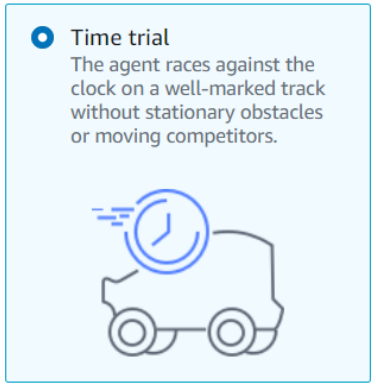
Log details for quick reference

The model description can have up to 255 characters.

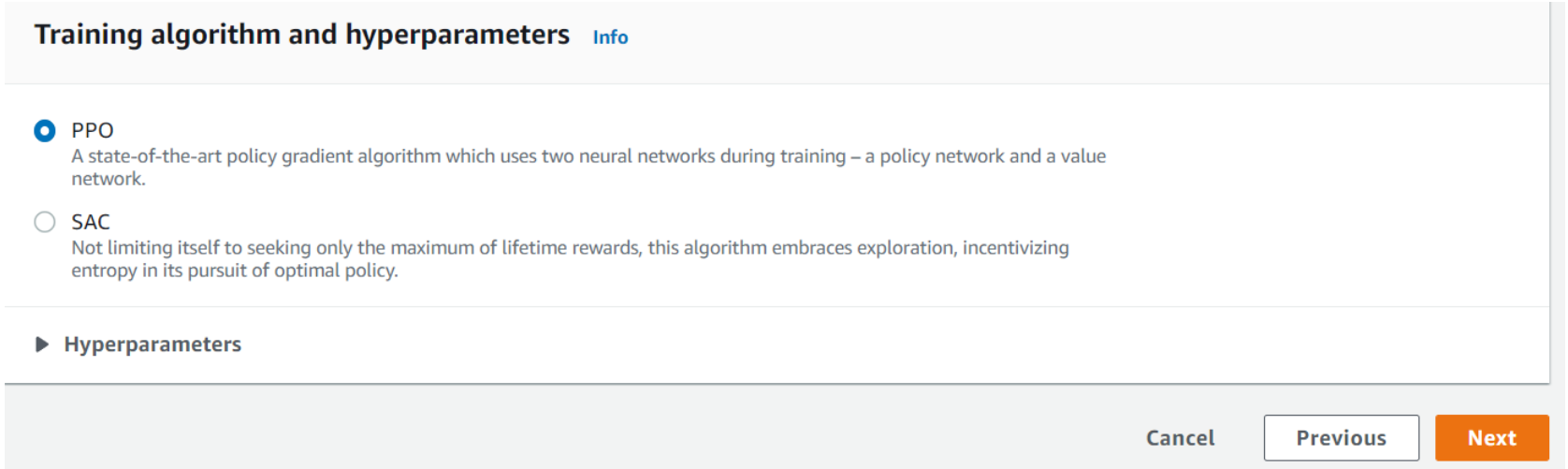


Your models- Choose race type and training algorithm

- **Race type**

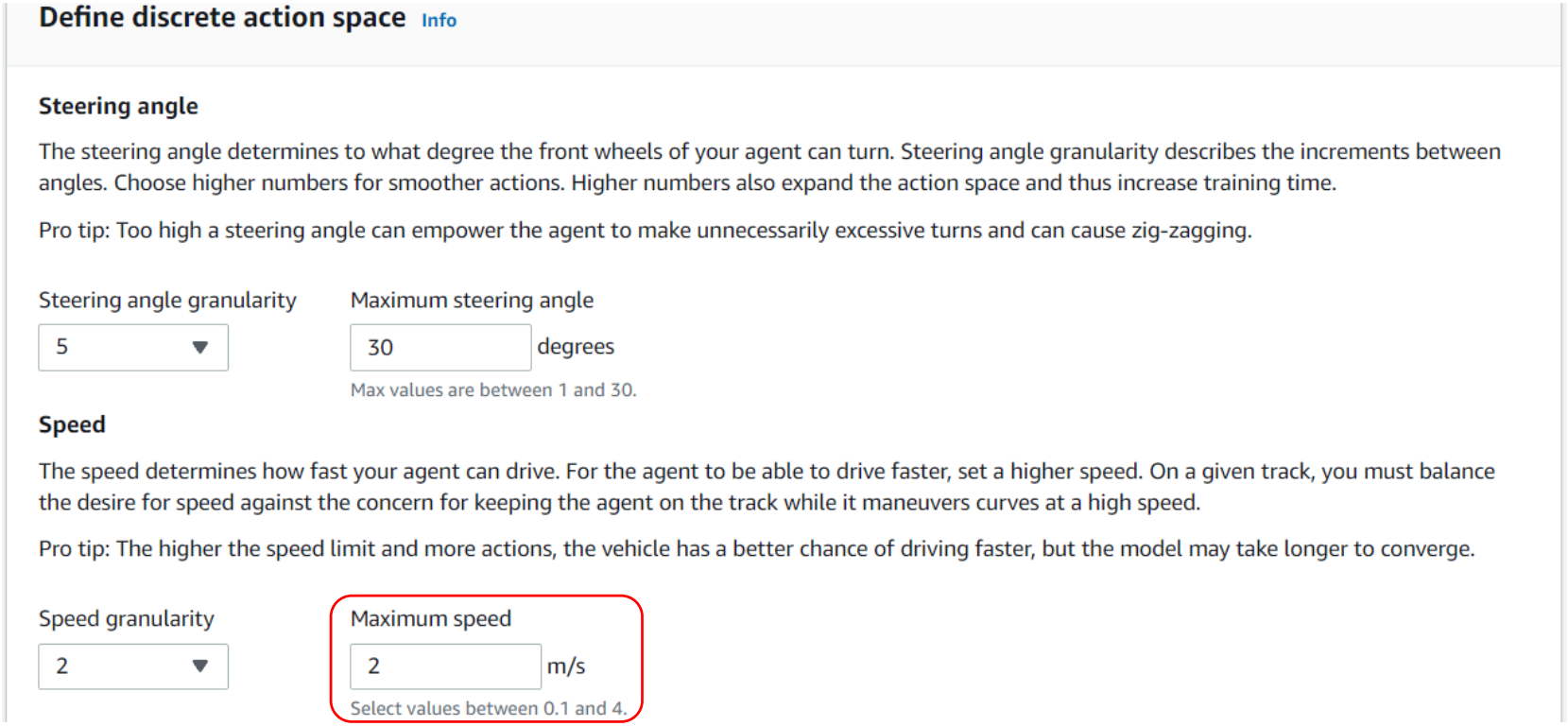


- **Training algorithm and hyperparameters- Choose PPO**



Your models- Define action space

- **Select action space- Discrete action space**
- **Define discrete action space**

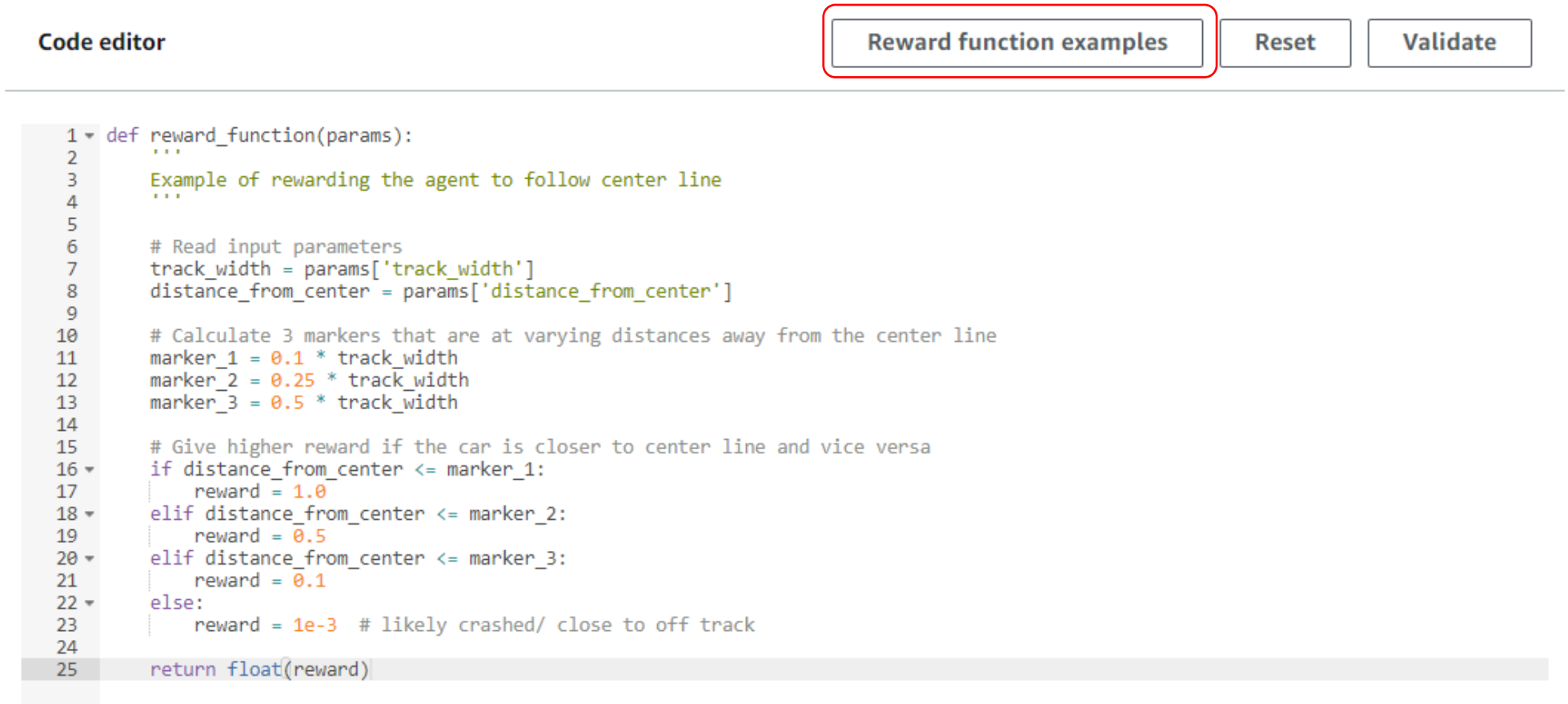


Your models- Choose vehicle

- **Choose The Original DeepRacer**

Your models- Customize reward function

- **Use Reward function examples to run your first model**



Your models- Customize reward function

- **Time trial - follow the center line (Default)**

This example determines how far away the agent is from the center line and gives higher reward if it is closer to the center of the track. It will incentivize the agent to closely follow the center line.

- **Time trial - stay inside the two borders**

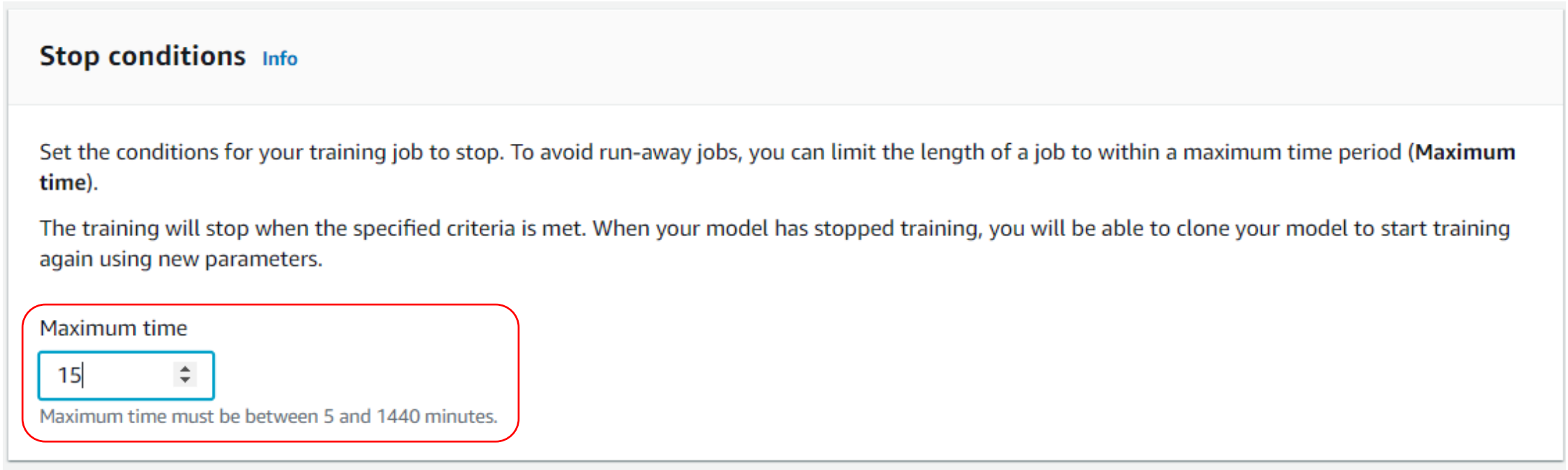
This example simply gives high rewards if the agent stays inside the borders and lets the agent figure out what is the best path to finish a lap. It is easy to program and understand, but will be likely to take longer time to converge.

- **Time trial - prevent zig-zag**

This example incentivizes the agent to follow the center line but penalizes with lower reward if it steers too much, which will help prevent zig-zag behavior. The agent will learn to drive smoothly in the simulator and likely display the same behavior when deployed in the physical vehicle.

Your models- Customize reward function

- **Stop conditions**



AWS DeepRacer Reward Function Examples (Link)

Example 1: Follow the Center Line in Time Trials

This example determines how far away the agent is from the center line, and gives higher reward if it is closer to the center of the track, encouraging the agent to closely follow the center line.

```
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
```

Example 2: Stay Inside the Two Borders in Time Trials

This example simply gives high rewards if the agent stays inside the borders, and let the agent figure out what is the best path to finish a lap. It is easy to program and understand, but likely takes longer to converge.

```
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
```

Example 3: Prevent Zig-Zag in Time Trials

This example incentivizes the agent to follow the center line but penalizes with lower reward if it steers too much, which helps prevent zig-zag behavior. The agent learns to drive smoothly in the simulator and likely keeps the same behavior when deployed in the physical vehicle.

```
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)
```