Now, let's create a new event from the command line:

```
$ curl -X 'POST' \
  'http://0.0.0.0:8080/event/new' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXV
CJ9.eyJ1c2VyIjoicmVhZGVyQHBhY2t0LmNvbSIsImV4cGlyZXMiOjE2NTA4Mjk
xODMuNTg3NjAyfQ.MOXjI5GXnyzGNftdlxDGyM119_L11uPq8yCxBHepf04' \
  -H 'Content-Type: application/json' \
  -d '{
  "title": "FastAPI Book Launch CLI",
  "image": "https://linktomyimage.com/image.png",
  "description": "We will be discussing the contents of the
  FastAPI book in this event.Ensure to come with your own
  copy to win gifts!",
  "tags": [
    "python",
    "fastapi",
    "book",
    "launch"
  ],
  "location": "Google Meet"
}'
```

In the request sent here, the `Authorization: Bearer` header is sent as well to inform the application that we are authorized to perform this action. The response gotten is the following:

```
{ "message": "Event created successfully" }
```

If we try to create an event without passing the authorization header with a valid token, an `HTTP 401 Unauthorized` error is returned:

```
$ curl -X 'POST' \
  'http://0.0.0.0:8080/event/new' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "title": "FastAPI BookLaunch",
```

```
  "image": "https://linktomyimage.com/image.png",
  "description": "We will be discussing the contents of the
  FastAPI book in this event.Ensure to come with your own
  copy to win gifts!",
  "tags": [
    "python",
    "fastapi",
    "book",
    "launch"
  ],
  "location": "Google Meet"
}'
```

Here's the response:

```
$ {
  "detail": "Not authenticated"
}
```

Now that we have successfully protected the routes, let's update the protected routes as follows:

- POST route: Add the event created to the list of events owned by the user.
- UPDATE route: Modify the route to ensure only the event created by the user can be updated.
- DELETE route: Modify the route to ensure only the event created by the user can be deleted.

In the previous section, we successfully injected the authentication dependencies to our route operations. To easily identify events and prevent a user from deleting another user's event, we'll update the event document class as well as the routes.

Updating the event document class and routes

Add the `creator` field to the `Event` document class in `models/events.py`:

```
class Event(Document):
    creator: Optional[str]
```

This field will enable us to restrict the operations performed on an event to the user alone.

Next, let's modify the POST route to update the `creator` field when creating a new event in `routes/events.py`:

```
@event_router.post("/new")
async def create_event(body: Event, user: str =
Depends(authenticate)) -> dict:
    body.creator = user
    await event_database.save(body)
    return {
        "message": "Event created successfully"
    }
```

In the preceding code block, we have updated the POST route to add the current user's email as the creator of the event. If you create a new event, the event is stored with the creator's email:

```
$ curl -X 'POST' \
  'http://0.0.0.0:8080/event/new' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXV
CJ9.eyJ1c2VyIjoicmVhZGVyQHBhY2t0LmNvbSIsImV4cGlyZXMiOjE2NTA4MzI
5NjQuMTU3MjQ4fQ.RxR1TYMx91JtVMNzYcT7718xXWX7skTCfWbnJxyf6fU' \
  -H 'Content-Type: application/json' \
  -d '{
  "title": "FastAPI Book Launch",
  "image": "https://linktomyimage.com/image.png",
  "description": "We will be discussing the contents of the
  FastAPI book in this event.Ensure to come with your own
  copy to win gifts!",
  "tags": [
    "python",
    "fastapi",
    "book",
    "launch"
  ],
  "location": "Google Meet"
```

```
}'
```

The response returned from the request above is:

```
{
  "message": "Event created successfully"
}
```

Next, let's retrieve the list of events stored in the database:

```
$ curl -X 'GET' \
  'http://0.0.0.0:8080/event/' \
  -H 'accept: application/json'
```

The response from the request above is:

```
[
  {
    "_id": "6265a807e0c8daefb72261ea",
    "creator": "reader@packt.com",
    "title": "FastAPI BookLaunch",
    "image": "https://linktomyimage.com/image.png",
    "description": "We will be discussing the contents of the
    FastAPI book in this event.Ensure to come with your own
    copy to win gifts!",
    "tags": [
      "python",
      "fastapi",
      "book",
      "launch"
    ],
    "location": "Google Meet"
  },
]
```

Next, let's update the UPDATE route:

```
@event_router.put("/{id}", response_model=Event)
async def update_event(id: PydanticObjectId, body: EventUpdate,
```

```
user: str = Depends(authenticate)) -> Event:
    event = await event_database.get(id)
    if event.creator != user:
        raise HTTPException(
            status_code=status.HTTP_400_BAD_REQUEST,
            detail="Operation not allowed"
        )
```

In the preceding code block, the route function checks whether the current user can edit an event before proceeding, otherwise, it raises an HTTP 400 bad request exception. Here's an example using a different user:

```
$ curl -X 'PUT' \
  'http://0.0.0.0:8080/event/6265a83fc823a3c912830074' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVC
J9.eyJ1c2VyIjoiZmFzdGFwaUBwYWNrdC5jb20iLCJleHBpcmVzIjoxNjUwODM
zOTc2LjI2NzgzMX0.MMRT6pwEDBVHTU5C1a6MV8j9wCfWhqbza9NBpZz08xE' \
  -H 'Content-Type: application/json' \
  -d '{
  "title": "FastAPI Book Launch"
}'
```

Here's the response:

```
{
  "detail": "Operation not allowed"
}
```

Lastly, let's update the DELETE route:

```
@event_router.delete("/{id}")
async def delete_event(id: PydanticObjectId, user: str =
Depends(authenticate)):
    event = await event_database.get(id)
    if event.creator != user:
        raise HTTPException(
```