PrefectHQ / **prefect** Public

Code Issues 804 Pull requests 41 Discussions Actions Security an

Commit `f8afeca`

3 people authored on Jan 27 · 60 / 62 · Verified

```
Add authentication to /api/events/in WebSocket endpoint (#20372)

Co-authored-by: Devin AI <158243242+devin-ai-integration[bot]@users.noreply.github.com>
Co-authored-by: alex.s@prefect.io <ajstreed1@gmail.com>
Co-authored-by: Claude Opus 4.5 <noreply@anthropic.com>
```

`main` · prefect-sqlalchemy-0.6.1 ••• 3.6.14.dev5

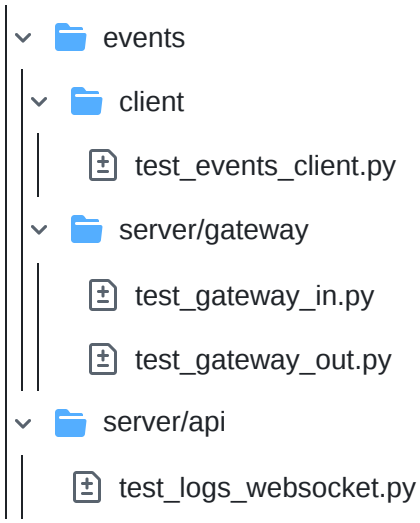
1 parent `e519a2e` commit `f8afeca`

9 files changed +280 -30

Top

Filter files…

- src/prefect
  - events
    - clients.py
  - server
    - api
      - events.py
    - utilities
      - subscriptions.py
  - testing
    - fixtures.py
- tests
  - _internal
    - test_websockets.py



Search within code

src/prefect/events/clients.py

```
@@ -33,12 +33,12 @@
33  33  from prefect.events import Event
34  34  from prefect.logging import get_logger
35  35  from prefect.settings import (
36      -       PREFECT_API_AUTH_STRING,
37  36          PREFECT_API_KEY,
38  37          PREFECT_API_URL,
39  38          PREFECT_CLOUD_API_URL,
40  39          PREFECT_DEBUG_MODE,
41  40          PREFECT_SERVER_ALLOW_EPHEMERAL_MODE,
    41  +       get_current_settings,
42  42  )
43  43
44  44  if TYPE_CHECKING:
@@ -260,6 +260,7 @@ class PrefectEventsClient(EventsClient):
260 260
261 261          _websocket: Optional[ClientConnection]
262 262          _unconfirmed_events: List[Event]
    263 +       _auth_token: Optional[str]
263 264
264 265          def __init__(
265 266              self,
@@ -281,8 +282,15 @@ def __init__(
281 282                      "api_url must be provided or set in the Prefect configuration"
```

```
282  283                )
283  284
     285  +         auth_string = get_current_settings().api.auth_string
     286  +         self._auth_token = (
     287  +             auth_string.get_secret_value() if auth_string is not None else None
     288  +         )
284  289          self._events_socket_url = events_in_socket_from_api_url(api_url)
285       -         self._connect = websocket_connect(self._events_socket_url)
     290  +         self._connect = websocket_connect(
     291  +             self._events_socket_url,
     292  +             subprotocols=[Subprotocol("prefect")],
     293  +         )
286  294          self._websocket = None
287  295          self._reconnection_attempts = reconnection_attempts
288  296          self._unconfirmed_events = []
     ↓ ↑  @@ -374,6 +382,28 @@ async def _reconnect(self) -> None:
374  382                )
375  383                raise
376  384
     385  +         logger.debug("Authenticating...")
     386  +         await self._websocket.send(
     387  +             orjson.dumps({"type": "auth", "token": self._auth_token}).decode()
     388  +         )
     389  +
     390  +         try:
     391  +             message: Dict[str, Any] = orjson.loads(await
          self._websocket.recv())
     392  +             logger.debug("Auth result: %s", message)
     393  +             assert message["type"] == "auth_success", message.get("reason", "")
     394  +         except AssertionError as e:
     395  +             raise Exception(
     396  +                 "Unable to authenticate to the event stream. Please ensure the
          "
     397  +                 "provided auth_token you are using is valid for this
          environment. "
     398  +                 f"Reason: {e.args[0]}"
     399  +             )
     400  +         except ConnectionClosedError as e:
     401  +             reason = getattr(e.rcvd, "reason", None)
```

```
      402 +             msg = "Unable to authenticate to the event stream. Please ensure
          the "
      403 +             msg += "provided auth_token you are using is valid for this
          environment. "
      404 +             msg += f"Reason: {reason}" if reason else ""
      405 +             raise Exception(msg) from e
      406 +
377   407             events_to_resend = self._unconfirmed_events
378   408             logger.debug("Resending %s unconfirmed events.", len(events_to_resend))
379   409             # Clear the unconfirmed events here, because they are going back
          through emit
          @@ -566,7 +596,10 @@ def __init__(
566   596                     the client should attempt to reconnect
567   597             """
568   598             self._api_key = None
569       -         self._auth_token = PREFECT_API_AUTH_STRING.value()
      599 +         auth_string = get_current_settings().api.auth_string
      600 +         self._auth_token = (
      601 +             auth_string.get_secret_value() if auth_string is not None else None
      602 +         )
570   603
571   604             if not api_url:
572   605                 api_url = cast(str, PREFECT_API_URL.value())
```

src/prefect/server/api/events.py

```
          @@ -64,8 +64,9 @@ async def create_events(
64    64    @router.websocket("/in")
65    65    async def stream_events_in(websocket: WebSocket) -> None:
66    66        """Open a WebSocket to stream incoming Events"""
67       -
68       -     await websocket.accept()
      67 +     websocket = await subscriptions.accept_prefect_socket(websocket)
      68 +     if not websocket:
      69 +         return
69    70
70    71        try:
71    72            async with messaging.create_event_publisher() as publisher:
```

src/prefect/server/utilities/subscriptions.py

```
@@ -18,9 +18,32 @@

async def accept_prefect_socket(websocket: WebSocket) -> Optional[WebSocket]:
    subprotocols = websocket.headers.get("Sec-WebSocket-Protocol",
"").split(",")
-   if "prefect" not in subprotocols:
-       return await websocket.close(WS_1002_PROTOCOL_ERROR)
+   has_prefect_subprotocol = "prefect" in subprotocols
+
+   auth_setting = (
+       auth_setting_secret.get_secret_value()
+       if (auth_setting_secret :=
get_current_settings().server.api.auth_string)
+       else None
+   )
+
+   # If client doesn't send "prefect" subprotocol:
+   # - Reject if auth is configured (security requirement)
+   # - Accept in legacy mode if auth is not configured (backward compatibility)
+   if not has_prefect_subprotocol:
+       if auth_setting:
+           logger.warning(
+               "WebSocket connection rejected: 'prefect' subprotocol required
when auth is configured"
+           )
+           return await websocket.close(WS_1002_PROTOCOL_ERROR)
+       else:
+           # Legacy mode: accept without auth handshake for old clients
+           logger.debug(
+               "Accepting WebSocket in legacy mode (no 'prefect' subprotocol)"
+           )
+           await websocket.accept()
+           return websocket

+   # New protocol: client sent "prefect" subprotocol, perform auth handshake
    await websocket.accept(subprotocol="prefect")

    try:
```

```
@@ -31,12 +54,6 @@ async def accept_prefect_socket(websocket: WebSocket) ->
Optional[WebSocket]:
31 54            # The protocol requires receiving an auth message for compatibility
32 55            # with Prefect Cloud, even if server-side auth is not configured.
33 56            message = await websocket.receive_json()
34    -
35    -        auth_setting = (
36    -            auth_setting_secret.get_secret_value()
37    -            if (auth_setting_secret := get_current_settings().server.api.auth_string)
38    -            else None
39    -        )
40 57            logger.debug(
41 58                f"PREFECT_SERVER_API_AUTH_STRING setting: {'*' * len(auth_setting) if auth_setting else 'Not set'}"
42 59            )
```

src/prefect/testing/fixtures.py

```
@@ -303,6 +303,7 @@ class Puppeteer:
303 303        outgoing_events: List[Event]
304 304
305 305        def __init__(self):
306     +          self.token = None
306 307            self.hard_auth_failure = False
307 308            self.refuse_any_further_connections = False
308 309            self.hard_disconnect_after = None
@@ -340,6 +341,22 @@ async def handler(socket: ServerConnection) -> None:
340 341                await outgoing_events(socket)
341 342
342 343        async def incoming_events(socket: ServerConnection):
344     +          # 1. authentication (always required, matching server behavior)
345     +          auth_message = json.loads(await socket.recv())
346     +
347     +          assert auth_message["type"] == "auth"
348     +          recorder.token = auth_message["token"]
349     +          if puppeteer.token is not None and puppeteer.token != recorder.token:
350     +              if not puppeteer.hard_auth_failure:
351     +                  await socket.send(
```

```
                                    json.dumps({"type": "auth_failure", "reason": "nope"})
                )
            await socket.close(WS_1008_POLICY_VIOLATION)
            return

        await socket.send(json.dumps({"type": "auth_success"}))

        # 2. receive events
        while True:
            try:
                message = await socket.recv()
```

```
@@ -388,7 +405,20 @@ async def outgoing_events(socket: ServerConnection):
                puppeteer.hard_disconnect_after = None
                raise ValueError("zonk")

-    async with serve(handler, host="localhost", port=unused_tcp_port) as server:
+    def select_subprotocol(
+        connection: ServerConnection, subprotocols: list[str]
+    ) -> str | None:
+        # Accept the "prefect" subprotocol if requested
+        if "prefect" in subprotocols:
+            return "prefect"
+        return None
+
+    async with serve(
+        handler,
+        host="localhost",
+        port=unused_tcp_port,
+        select_subprotocol=select_subprotocol,
+    ) as server:
            yield server
```

tests/_internal/test_websockets.py

```
@@ -1,8 +1,10 @@
+ import json
```

```
1   2     import ssl
2   3     import warnings
3   4     from unittest.mock import patch
4   5
5   6     from websockets.asyncio.client import connect
    7   + from websockets.protocol import Subprotocol
6   8
7   9     from prefect._internal.websockets import (
8   10        create_ssl_context_for_websocket,
      @@ -164,9 +166,18 @@ async def
      test_websocket_custom_headers_with_websocket_connect(hosted_api_server
164 166       custom_headers = {"X-Custom-Header": "test-value"}
165 167
166 168       with temporary_settings({PREFECT_CLIENT_CUSTOM_HEADERS: custom_headers}):
167     -         connector =
      websocket_connect(events_in_socket_from_api_url(hosted_api_server))
        169   +         connector = websocket_connect(
        170   +             events_in_socket_from_api_url(hosted_api_server),
        171   +             subprotocols=[Subprotocol("prefect")],
        172   +         )
168 173           # Make sure we can connect to the websocket successfully with the
      custom headers
        174   +         # The /events/in endpoint requires the "prefect" subprotocol and auth
      handshake
169 175           async with connector as websocket:
170 176               pong = await websocket.ping()
171 177               await pong
        178   +
        179   +             # Send auth message (required by the server)
        180   +             await websocket.send(json.dumps({"type": "auth", "token": None}))
        181   +             auth_response = json.loads(await websocket.recv())
        182   +             assert auth_response["type"] == "auth_success"
172 183               # If we get here, the connection worked with custom headers
```

tests/events/client/test_events_client.py

```
      @@ -525,7 +525,7 @@ async def send(self, data):
525 525               pass
526 526
527 527           async def recv(self):
528     -             pass
```

```
      528 +             return '{"type": "auth_success"}'
529   529
530   530         class MockConnect:
531   531             def __init__(self):
          @@ -546,7 +546,6 @@ def mock_connect(*args, **kwargs):
546   546
547   547         monkeypatch.setattr("prefect.events.clients.websocket_connect",
          mock_connect)
548   548
549       -     # Should succeed after retrying
550   549         async with PrefectEventsClient("ws://localhost") as client:
551   550             assert client._websocket is not None
552   551
          @@ -640,6 +639,9 @@ async def ping(self):
640   639             async def send(self, data):
641   640                 pass
642   641
      642 +         async def recv(self):
      643 +             return '{"type": "auth_success"}'
      644 +
643   645         class MockConnect:
644   646             def __init__(self):
645   647                 self.connection = None
```

Comments 0

Please sign in to comment.