```
if USE_INTROSPECTION:
    # Use introspection to validate the token
    introspection_result = await introspect_token(token)
    if not introspection_result.get("active"):
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED, detail="Token is inactive or invalid"
        )
    # Extract user data and scopes from introspection result
    user_data = {
        "sub": introspection_result["sub"],
        "username": introspection_result.get("username", introspection_result["sub"]),
    }

    # Get scopes
    token_scopes = introspection_result.get("scope", "").split()

    # Validate the 'aud' claim
    if introspection_result.get("aud") != config('OKTA_AUDIENCE'):
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED, detail="Invalid audience"
        )
```

Local validation with Authlib

```
else:
    # Perform local verification
    jwks_data = await get_jwks()
    rsa_key = None
    for key in jwks_data["keys"]:
        rsa_key = JsonWebKey.import_key(key)
        break
    # Decode the token
    payload = jwt.decode(
        token, rsa_key, claims_options={"exp": {"essential": True}}
    )
    # Validate the 'aud' claim
    if payload.get("aud") ≠ config('OKTA_AUDIENCE'):
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED, detail="Invalid audience"
        )
    # Validate the 'iss' claim
    if payload.get("iss") ≠ AUTHORIZATION_SERVER_URL:
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED, detail="Invalid issuer"
        )
```