[Homepage](https://medium.com/)

[Technology & Learning](https://medium.com/technology-learning?source=logo-lo_454e93ab1f34---f8c9bb80f276)

Follow

[Sign in / Sign up](https://medium.com/m/signin?redirect=https%3A%2F%2Fmedium.com%2Ftechnology-learning%2Fhow-we-solved-authentication-and-authorization-in-our-microservice-architecture-994539d1b6e6)

[Go to the profile of OKONKWO VINCENT IKEM](https://medium.com/@ikem?source=post_header_lockup)

[OKONKWO VINCENT IKEM](https://medium.com/@ikem?source=post_header_lockup)Follow

A journeyman in the field of software craftmanship. A coach. CEO core infrastructure at @ andela. Passionate about education and lifelong learning.

Apr 24

**How We Solved Authentication and Authorization in Our Microservice Architecture**

At [Andela](https://andela.com/what-we-do/" \t "_blank), we have multiple internal apps built by the internal Engineering teams used to manage internal process and to make us more efficient. As time passed, the number of apps increased and each app had to evolve alongside the passing of time to enable us handle our growth. We started experiencing a number of pain points(documented [here](https://medium.com/technology-learning/microservices-migration-lessons-learned-f84166ec7eb0)) which forced us to look towards microservices for salvation.

If you want to know more about our microservice journey thus far, I recommend you check out [Scalable Architecture with EventSourcing and CQRS](https://medium.com/technology-learning/event-sourcing-and-cqrs-a-look-at-kafka-e0c1b90d17d8), [Antifragile Microservice](https://medium.com/technology-learning/building-out-antifragile-microservice-andela-design-consideration-d6e03a185d6a" \t "_blank) and [From Monolith to Microservices](https://medium.com/@briceicle/migrating-from-a-monolith-to-a-microservices-architecture-99cecf8af366)blogposts.

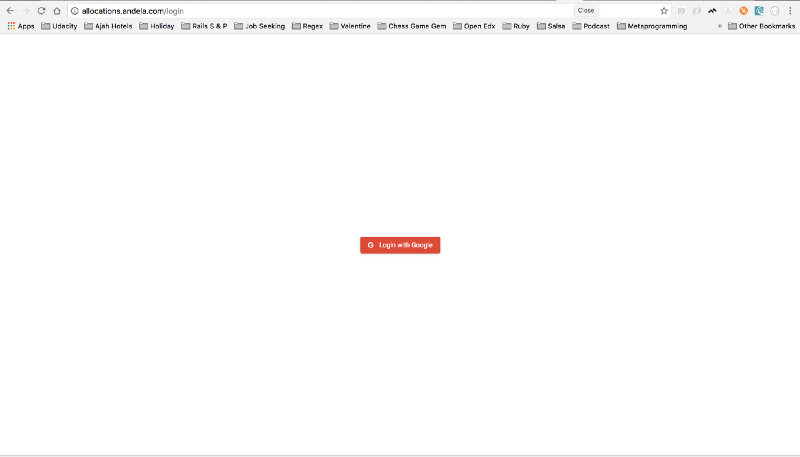
**Building Blocks of our authentication/authorization layer**

Our architecture has a number of building blocks working together to achieve a robust authentication/authorization layer.



**FrontEnd Apps**

Each app(skilltree, kaizen, pulse, allocations) is independent and written in different frontend framework. The login page for each app is similar and simple. Below is the login page for allocations app.



To login, the user just clicks the Login with Google button which is a link to `<http://api-prod.andela.com/login?redirect_url=http://allocations.andela.com>`. Once the user clicks the button, the login endpoint of the api gateway picks it up, performs all the necessary magic, logs the user in and redirects the user to the calling application.

**Token Based Authentication**

In a monolith, it’s ok for it to be built as a stateful application. Hence, session based authentication works really well. However, that’s not the case with microservices, since you need to route requests to multiple independent services. To maintain statelessness in our system, we opted to use token authentication. We packaged user claims in the jwt. JSON Web Token(jwt) is an open, industry standard [**RFC 7519**](https://tools.ietf.org/html/rfc7519) method for representing claims securely between two parties.

When a user logs in successfully, a jwt is returned. On subsequent requests, the user must attach this token to each request. This is needed so that the api gateway can establish the identity of the user and extract the user’s claims from the token.

**Single SignON(SSO)**

SSO simply means login, just once to a suite of independent applications. In our case, once you login to one application(eg Skilltree), you won’t need to login again when you attempt to access another application(eg Pulse) since you will be automatically signed in. With SSO, our users won’t go through the hassle of signing in each time they want to use one of our internal apps.

The way SSO is implemented in our system is simple. When a user logs in for the first time from any frontend app, a cookie called jwt-token gets created on the api-gateway. The cookie’s domain is .andela.com and hence accessible to all andela.com subdomain. When a request is made from any of the frontend apps to the api gateway, we extract the cookie named jwt-token if set. If not set, we assume the user is not logged in and return a 401status code from the api-gateway.

NOTE: All our internal apps are hosted in andela.com subdomain eg skilltree.andela.com, pulse.andela.com, allocations.andela.com etc.

**Mobile Apps**

Since a mobile app is obviously not on andela.com subdomain, the api-gateway supports passing the jwt via Authorization header as Bearer Tokens. The auth flow for mobile apps is a little bit different from web apps. Since we are using Google oauth, the oauth flow happens in the mobile app. Once the user is successfully authenticated via google, we use the Google access tokens to exchange for a jwt from the api gateway.

**Third Party Apps**

Our system also supports api-token authentication. This is necessary because it’s not every time a user is involved in the authentication process. You might just need a third party app to have access to some data. Each user has an api-token they can use to access the application from a third party app. Also, we can create a service account that only has access to our system through the api-tokens.

**Authentication**

The api gateway is the middleman between the frontend apps and the suite of microservices. It’s responsible for generating the jwt and hence authentication. It achieves this by communicating with authorization and users service. The api gateway is written in golang and the auth logic was extracted out as a package and mounted as a middleware. You can find a snippet of the authentication code [here](https://gist.github.com/codesword/1b997a259d2509c3b0ea7f087c9b5466).

handleOauth2Callback is the method invoked by google callback url. The method exchanges the code returned for a google access token. Using this token it retrieves the user’s profile information. This profile info is packaged as a user struct and used to generate the jwt. Generating the jwt involves making a call to users service FindOrCreate endpoint and FetchPermissions endpoint of authorization service. The information received from both endpoints will be the claims in the jwt. Once the information is retrieved, a cookie called jwt-token will be set with the jwt as the value provided the calling app is on andela.com subdomain. Otherwise, the token is returned back to the user as a query string.

**Claims**

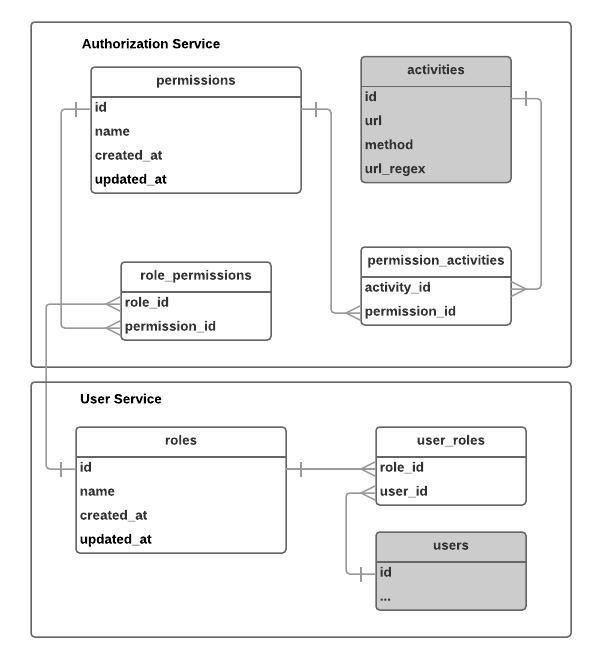
A number of user information are packaged as claims in the jwt. Below are the claims in a sample jwt.

**Authorization**

As we moved from monoliths to microservices, we needed to centralize our authorization effort by creating an authorization service. Authorization in our systems is purely permission based. The permissions are used to restrict access to an api endpoint and also control users view on the frontend apps.

permissions in authorization service has many to many relationship with roles in user service. A permission has a many to many relationship with activities(endpoint). A user has many roles and hence many permissions. When a user makes a request to the api gateway, the request passes through the authorization middleware which extracts the jwt(from cookie or header), verifies that it’s valid and retrieves the permissions claim from the jwt. Afterwards, the Authorize endpoint of the authorization service is called with the permissions as well as the url and http verb of the called endpoint. The authorize endpoint essentially returns true if any of the user’s permission has access to the endpoint.

Below is the ER diagram of the authorization service showing relationship with user service.



The query that checks if a user is authorized or not looks like this.

SELECT COUNT(\*) FROM permission\_activities pa   
JOIN activities a ON pa.activity\_id = a.id   
WHERE a.method = :method AND :url ~ a.url\_regex AND pa.permission\_id IN(:permissionIds)

If this query returns any value greater than 0, then the user is authorized. From the query :method can be any of POST, GET, PUT, DELETE, PATCH. :url is the url of the endpoint the user is trying to access eg /api/v1/roles, or /api/v1/roles/some-id/users. The activities table has a regex field that enables matching to a wildcard. eg a url\_regex field can have value /api/v1/roles/[^/?#]+/users and hence will match /api/v1/roles/some-id/users. :permissionIds is an array of the permissions the user has access to. This permission list is retrieved from the user claims.

**Authorized Fields**

The above authorization setup works in most scenarios. However, there are some situations where different users have access to the same endpoint but the content they see are different i.e some users can see extra properties in the returned result. In this kind of scenario, authorization still happens as usual via the authorization service, however the microservice being called will still receive the users permissions as metadata and it will return specific fields based on the user’s permission.

**Conclusion**

Building a robust authentication/authorization system in a microservice architecture is not trivial. This is even more tricky when you have different applications consuming the same microservices. You also don’t want to start making modifications to code each time new endpoints are added or new applications are built.

Please share your experiences building out authorization in a microservice architecture in the comment section. I will love to hear from you.

*If you liked this, click the💚 below so other people will see this here on Medium. Also, if you have any question or observation, use the comment section to share your thoughts/questions.*

Thanks to [Scott Carleton](https://medium.com/@scotterc?source=post_page).

* [API](https://medium.com/tag/api?source=post)
* [Microservices](https://medium.com/tag/microservices?source=post)
* [Jwt](https://medium.com/tag/jwt?source=post)
* [Authentication](https://medium.com/tag/authentication?source=post)
* [Authorization](https://medium.com/tag/authorization?source=post)

319

16

 Follow

[Go to the profile of OKONKWO VINCENT IKEM](https://medium.com/@ikem?source=footer_card)

[**OKONKWO VINCENT IKEM**](https://medium.com/@ikem)

A journeyman in the field of software craftmanship. A coach. CEO core infrastructure at @ andela. Passionate about education and lifelong learning.

 Follow

[Technology & Learning](https://medium.com/technology-learning?source=footer_card)

[**Technology & Learning**](https://medium.com/technology-learning?source=footer_card)

Technology & Learning at Andela

[More from Technology & Learning](https://medium.com/technology-learning/prologue-designing-for-human-transformation-cd1fd7ac1ab1?source=placement_card_footer_grid---------0-41)

[Prologue: Designing for Human Transformation](https://medium.com/technology-learning/prologue-designing-for-human-transformation-cd1fd7ac1ab1?source=placement_card_footer_grid---------0-41)

[Go to the profile of Michael Rosenberg](https://medium.com/@writerosenberg)

[Michael Rosenberg](https://medium.com/@writerosenberg?source=placement_card_footer_grid---------0-41)

37

[Also tagged API](https://dev-blog.apollodata.com/tutorial-graphql-subscriptions-server-side-e51c32dc2951?source=placement_card_footer_grid---------1-43)

[Tutorial: GraphQL Subscriptions (server-side)](https://dev-blog.apollodata.com/tutorial-graphql-subscriptions-server-side-e51c32dc2951?source=placement_card_footer_grid---------1-43)

[Go to the profile of Shadaj Laddad](https://dev-blog.apollodata.com/@shadaj)

[Shadaj Laddad](https://dev-blog.apollodata.com/@shadaj?source=placement_card_footer_grid---------1-43)

52

[Related reads](https://dev-blog.apollodata.com/the-next-step-for-realtime-data-in-graphql-b564b72eb07b?source=placement_card_footer_grid---------2-40)

[The next step for realtime data in GraphQL](https://dev-blog.apollodata.com/the-next-step-for-realtime-data-in-graphql-b564b72eb07b?source=placement_card_footer_grid---------2-40)

[Go to the profile of Sashko Stubailo](https://dev-blog.apollodata.com/@stubailo)

[Sashko Stubailo](https://dev-blog.apollodata.com/@stubailo?source=placement_card_footer_grid---------2-40)

252

**Responses**

Write a response…

Conversation between [Alex Yang](https://medium.com/@AlexBoYang) and [OKONKWO VINCENT IKEM](https://medium.com/@ikem).

[](https://medium.com/@AlexBoYang)

[Alex Yang](https://medium.com/@AlexBoYang?source=responses---------0-----------)

[May 19](https://medium.com/@AlexBoYang/my-solutions-b06d52be6d59?source=responses---------0-----------)

[My solutions:](https://medium.com/@AlexBoYang/my-solutions-b06d52be6d59?source=responses---------0-----------)

[I used a API gateway + OAuth services to handle authentication. Client side gets a token from OAuth services and use that token to visit different microservices. The APIGateway will verify the token (valid, not expired or revoked) and get latest user information from user services (and cache). All other…](https://medium.com/@AlexBoYang/my-solutions-b06d52be6d59?source=responses---------0-----------)

Read more…

2

[1 response](https://medium.com/@AlexBoYang/my-solutions-b06d52be6d59?source=responses---------0-----------#--responses)

[Go to the profile of OKONKWO VINCENT IKEM](https://medium.com/@ikem)

[OKONKWO VINCENT IKEM](https://medium.com/@ikem?source=responses---------0-----------)

[May 19](https://medium.com/@ikem/that-makes-sense-c814f225419f?source=responses---------0-----------)

[That makes sense. Having a central authorization has it’s own complexities. Managing it at the microservice level is definitely a simpler and more straight forward approach. For us, the reason why we chose a central authorization is because we want to be able to create new permissions and map them to endpoints directly via a UI. This is especially…](https://medium.com/@ikem/that-makes-sense-c814f225419f?source=responses---------0-----------)

Read more…

2

Recommended by [OKONKWO VINCENT IKEM](https://medium.com/@ikem) (author)

[Go to the profile of Trevor Ewen](https://medium.com/@tewen)

[Trevor Ewen](https://medium.com/@tewen?source=responses---------1-31----------)

[May 18](https://medium.com/@tewen/shared-middleware-approach-a42e85855283?source=responses---------1-31----------)

[Shared middleware approach](https://medium.com/@tewen/shared-middleware-approach-a42e85855283?source=responses---------1-31----------)

[OKONKWO VINCENT IKEM](https://medium.com/@tewen/shared-middleware-approach-a42e85855283?source=responses---------1-31----------)

[We had a lot of luck with the shared middleware approach at my last client. These were node / express services, so it was fairly easy to keep the barrier to entry consistent across services.](https://medium.com/@tewen/shared-middleware-approach-a42e85855283?source=responses---------1-31----------)

Read more…

2

[1 response](https://medium.com/@tewen/shared-middleware-approach-a42e85855283?source=responses---------1-31----------#--responses)

Conversation with [OKONKWO VINCENT IKEM](https://medium.com/@ikem).

[Go to the profile of Kariuki](https://medium.com/@kariuki)

[Kariuki](https://medium.com/@kariuki?source=responses---------2-----------)

[Apr 25](https://medium.com/@kariuki/very-solid-article-d19876320c26?source=responses---------2-----------)

[Very solid article!](https://medium.com/@kariuki/very-solid-article-d19876320c26?source=responses---------2-----------)

[One thing that comes to mind when using JWT is that there is no valid way to invalidate a token after a user logs out.](https://medium.com/@kariuki/very-solid-article-d19876320c26?source=responses---------2-----------)

[I assume that the approach is to delete the JWT token from the browser’s local storage. But if I copied out the token before the user logs out, then effectively…](https://medium.com/@kariuki/very-solid-article-d19876320c26?source=responses---------2-----------)

Read more…

2

[6 responses](https://medium.com/@kariuki/very-solid-article-d19876320c26?source=responses---------2-----------#--responses)

[Go to the profile of OKONKWO VINCENT IKEM](https://medium.com/@ikem)

[OKONKWO VINCENT IKEM](https://medium.com/@ikem?source=responses---------2-----------)

[Apr 25](https://medium.com/@ikem/kariuki-thats-a-problem-we-haven-t-solved-e94cbee6c15c?source=responses---------2-----------)

[Kariuki that’s a problem we haven’t solved. I will like to hear on your opinion on this. Maybe possible approaches that can be taken.](https://medium.com/@ikem/kariuki-thats-a-problem-we-haven-t-solved-e94cbee6c15c?source=responses---------2-----------)

2

Conversation with [OKONKWO VINCENT IKEM](https://medium.com/@ikem).

[Go to the profile of Timo M. Staudinger](https://medium.com/@TimoSta)

[Timo M. Staudinger](https://medium.com/@TimoSta?source=responses---------3-----------)

[Apr 24](https://medium.com/@TimoSta/how-do-you-handle-access-restrictions-on-a-more-granular-level-e-g-404789b66bcd?source=responses---------3-----------)

[The activities table has a regex field that enables matching to a wildcard. eg a url\_regex field can have value /api/v1/roles/[^/?#]+/users and hence will match /api/v1/roles/some-id...](https://medium.com/technology-learning/how-we-solved-authentication-and-authorization-in-our-microservice-architecture-994539d1b6e6" \l "b006)

[How do you handle access restrictions on a more granular level, e.g. the user can POST his own profile at /api/v1/users/123, but not John’s at /api/v1/users/456? Would you have the micro service responsible for updating the user handle this?](https://medium.com/@TimoSta/how-do-you-handle-access-restrictions-on-a-more-granular-level-e-g-404789b66bcd?source=responses---------3-----------)

[1 response](https://medium.com/@TimoSta/how-do-you-handle-access-restrictions-on-a-more-granular-level-e-g-404789b66bcd?source=responses---------3-----------#--responses)

[Go to the profile of OKONKWO VINCENT IKEM](https://medium.com/@ikem)

[OKONKWO VINCENT IKEM](https://medium.com/@ikem?source=responses---------3-----------)

[Apr 25](https://medium.com/@ikem/thanks-timo-m-da2c37e5c7e9?source=responses---------3-----------)

[Thanks Timo M. Staudinger for this question. This is a problem we faced and solved. For a user accessing/modifying his own profile, we create a different route eg /api/v1/users/me. We retrieve the user id from the claims extracted from the logged in user jwt. This way, regular users only have access to /api/v1/users/me while admins has access to /api/v1/users/:id.](https://medium.com/@ikem/thanks-timo-m-da2c37e5c7e9?source=responses---------3-----------)

2

[1 response](https://medium.com/@ikem/thanks-timo-m-da2c37e5c7e9?source=responses---------3-----------#--responses)

[Go to the profile of Timo M. Staudinger](https://medium.com/@TimoSta)

[Timo M. Staudinger](https://medium.com/@TimoSta?source=responses---------3-----------)

[Apr 25](https://medium.com/@TimoSta/ah-i-see-a1e91eaa048d?source=responses---------3-----------)

[Ah, I see. How does this transfer to entities that are in a 1:n relationship with the user though? Say, a user creates posts, which he should be able to edit later on. Would you also use /api/v1/users/me/posts/1 instead of /api/v1/posts/1 to be able to distinguish between posts of the currently signed on user and and everyone else’s?](https://medium.com/@TimoSta/ah-i-see-a1e91eaa048d?source=responses---------3-----------)

[1 response](https://medium.com/@TimoSta/ah-i-see-a1e91eaa048d?source=responses---------3-----------#--responses)

[Go to the profile of OKONKWO VINCENT IKEM](https://medium.com/@ikem)

[OKONKWO VINCENT IKEM](https://medium.com/@ikem?source=responses---------3-----------)

[Apr 25](https://medium.com/@ikem/it-depends-203b69094653?source=responses---------3-----------)

[It depends. If a user can only edit post they have created, it will be controlled from 2 layers (authorization service and the microservice itself). The authorization service will check if the user has access to modify a post while the microservice will ensure that the user is modifying a post they own. This is possible because a post is always tied…](https://medium.com/@ikem/it-depends-203b69094653?source=responses---------3-----------)

Read more…

2

Conversation with [OKONKWO VINCENT IKEM](https://medium.com/@ikem).

[Go to the profile of Jade](https://medium.com/@QinLu_49957)

[Jade](https://medium.com/@QinLu_49957?source=responses---------4-----------)

[May 14](https://medium.com/@QinLu_49957/hi-i-am-the-editor-of-infoq-china-which-focuses-on-software-development-cbd9c17c28af?source=responses---------4-----------)

[Hi,I am the editor of InfoQ China which focuses on software development. We like this articles and plan to translate it.](https://medium.com/@QinLu_49957/hi-i-am-the-editor-of-infoq-china-which-focuses-on-software-development-cbd9c17c28af?source=responses---------4-----------)

[Before we translate it into Chinese and publish it on our website, I want to ask for your permission first! This translation version is provided for informational purposes only, and does not make any…](https://medium.com/@QinLu_49957/hi-i-am-the-editor-of-infoq-china-which-focuses-on-software-development-cbd9c17c28af?source=responses---------4-----------)

Read more…

[1 response](https://medium.com/@QinLu_49957/hi-i-am-the-editor-of-infoq-china-which-focuses-on-software-development-cbd9c17c28af?source=responses---------4-----------#--responses)

[Go to the profile of OKONKWO VINCENT IKEM](https://medium.com/@ikem)

[OKONKWO VINCENT IKEM](https://medium.com/@ikem?source=responses---------4-----------)

[May 15](https://medium.com/@ikem/jade-thanks-for-reaching-out-thats-fine-you-can-go-ahead-and-translate-publish-it-b52721be8edc?source=responses---------4-----------)

[Jade Thanks for reaching out. That’s fine. You can go ahead and translate/publish it.](https://medium.com/@ikem/jade-thanks-for-reaching-out-thats-fine-you-can-go-ahead-and-translate-publish-it-b52721be8edc?source=responses---------4-----------)

Conversation with [OKONKWO VINCENT IKEM](https://medium.com/@ikem).

[Go to the profile of Alex Sherwin](https://medium.com/@alex.sherwin)

[Alex Sherwin](https://medium.com/@alex.sherwin?source=responses---------5-----------)

Apr 25