AuthN and **AuthZ** at Cruise: Crawl, Walk, Run

Roman Porter and James Barclay
October 15, 2021

Setting the Stage...

- **~2000 employees**
- **~1500 engineers**
- **~150 services**

- **Majority of services in Kubernetes**
  - Microservice + monolith services
  - Spanning multiple clusters
  - Traffic is mostly HTTP-based

- **Using Vault for storing and managing secrets**

- **Using a cloud-based IdP for employee identity**

Three Types of Callers

Browser Clients (OIDC)

CLI Clients (JWT)

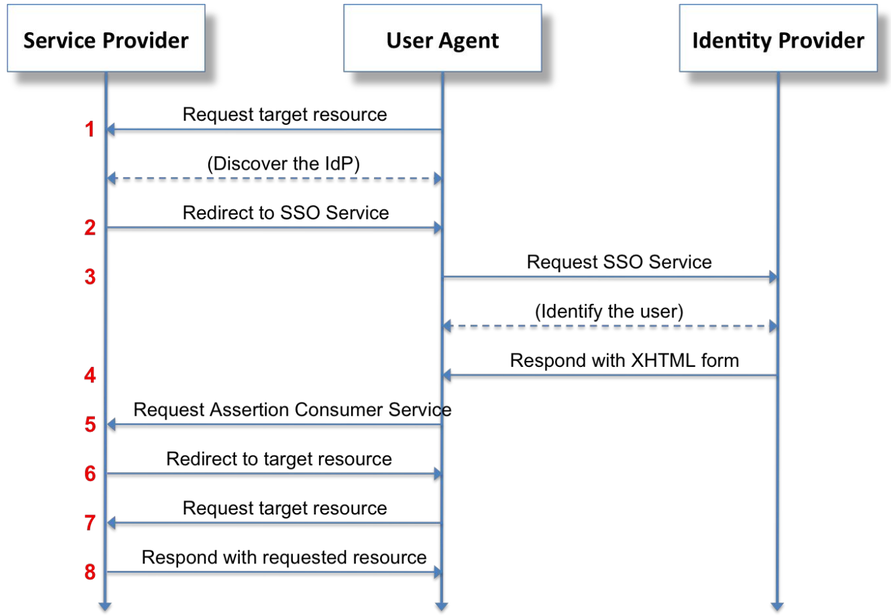
Service-to-Service Clients (JWT)

The Bad Old Days

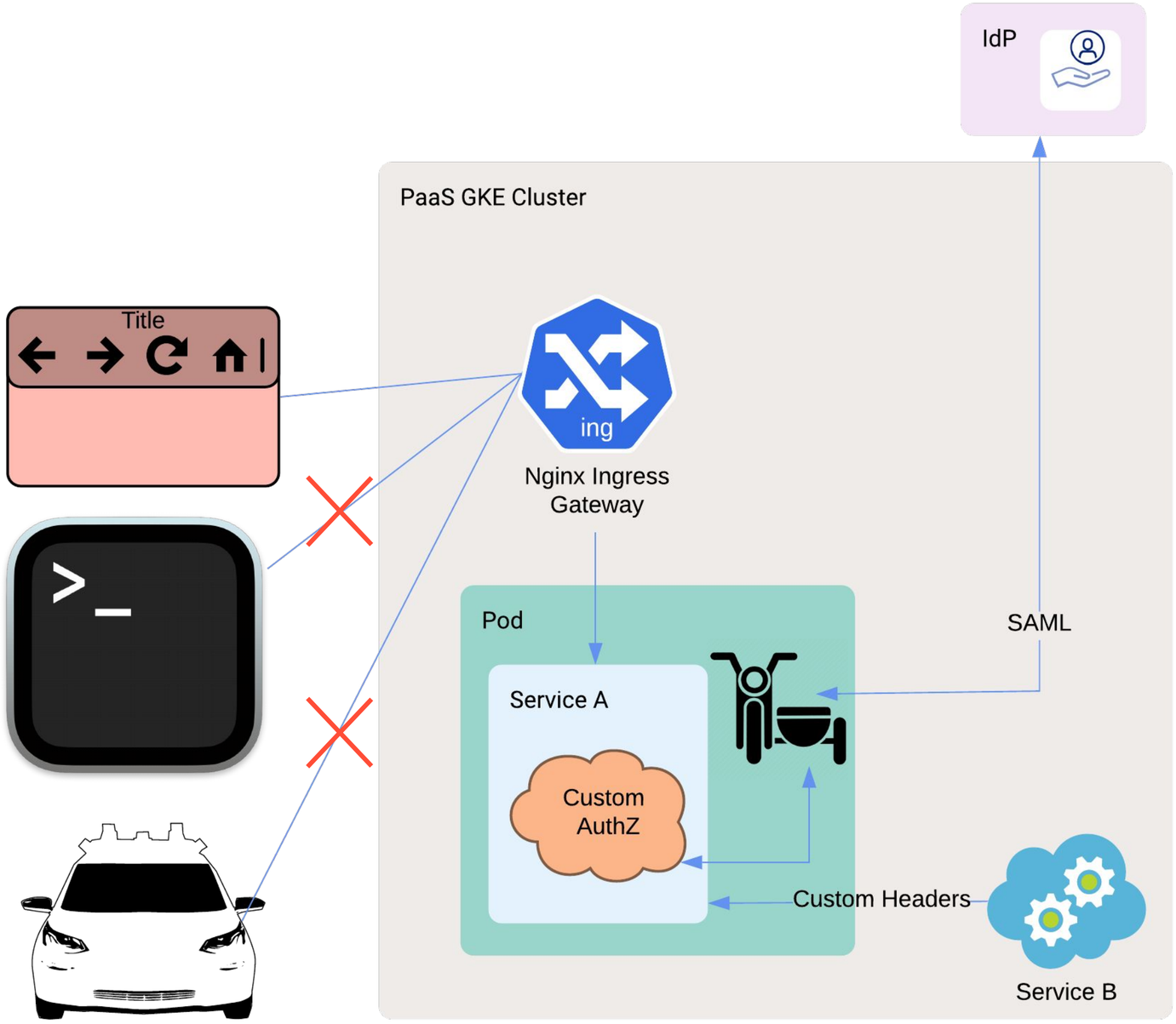
- Authentication and authorization • s p r a w l •
  - Teams handling authentication and authorization differently, with different IdPs
    - **OIDC** with IdP1
    - **OIDC** or **SAML** with IdP2
  - Naive implementations using static or shared credentials
  - Nothing at all? 😬

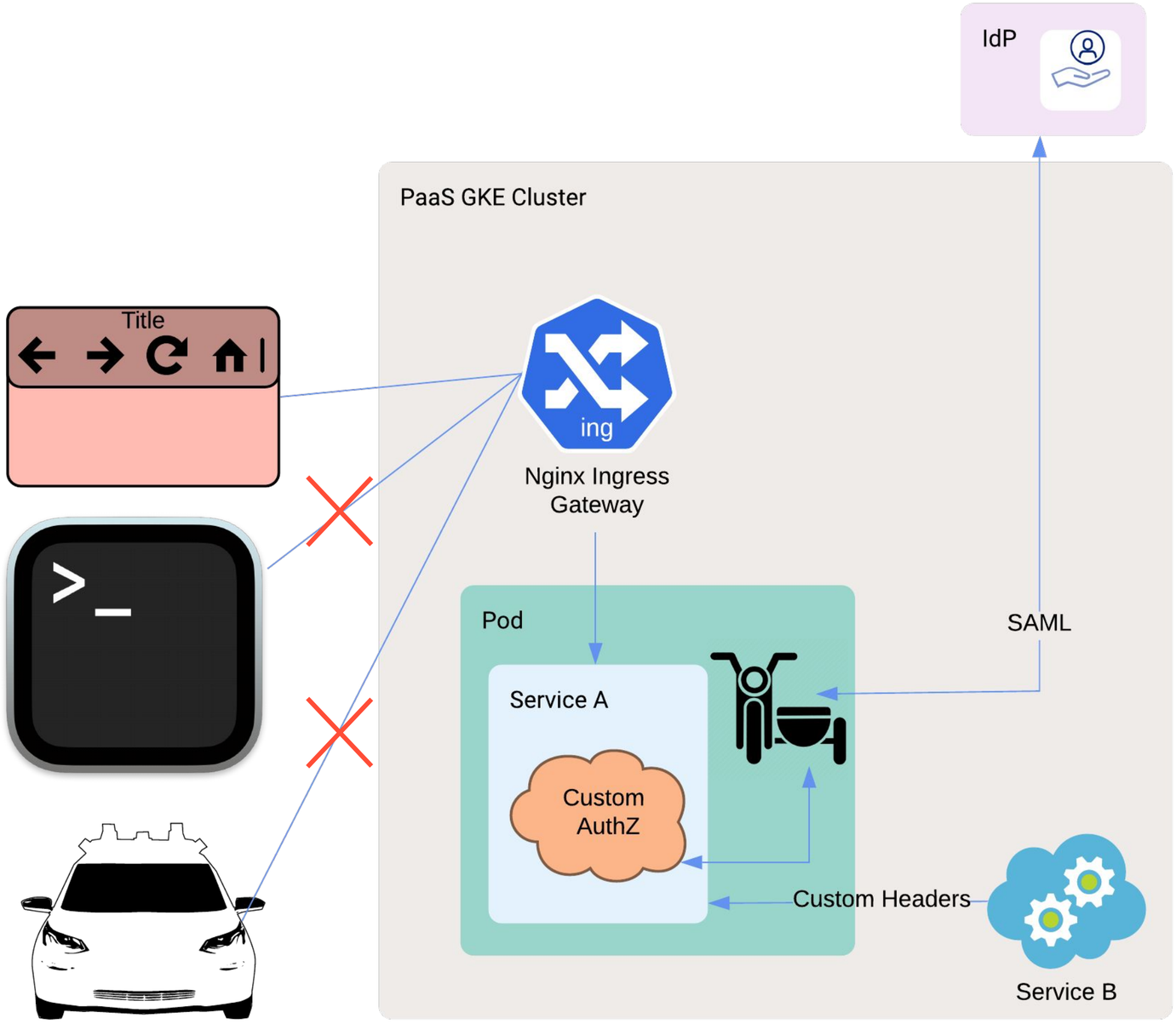
AuthN/Z Standardization

Phase I: Crawl



- Build a SAML authenticating proxy sidecar.
- Hope that people use it.

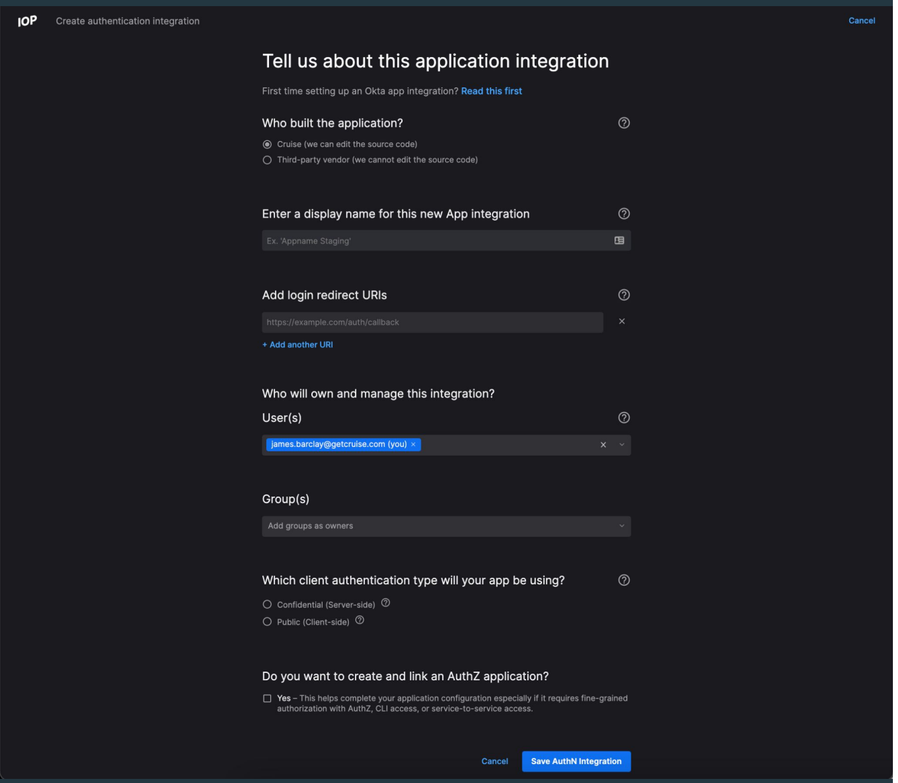

IdP
PaaS GKE Cluster
Title
ing
Nginx Ingress
Gateway
Pod
SAML
Service A
Custom
AuthZ
Custom Headers
Service B


IdP
PaaS GKE Cluster
Title
ing
Nginx Ingress
Gateway
>_
Pod
SAML
Service A
Custom
AuthZ
Custom Headers
Service B

- **Problems we had at the crawl stage**
  - Creating authentication (SAML) integrations was still **request-driven**.
  - Creating and modifying IdP groups was still **request-driven**.
  - Assigning access to authentication integrations was still **request-driven**.

AuthN/Z Standardization

Phase II: Walk

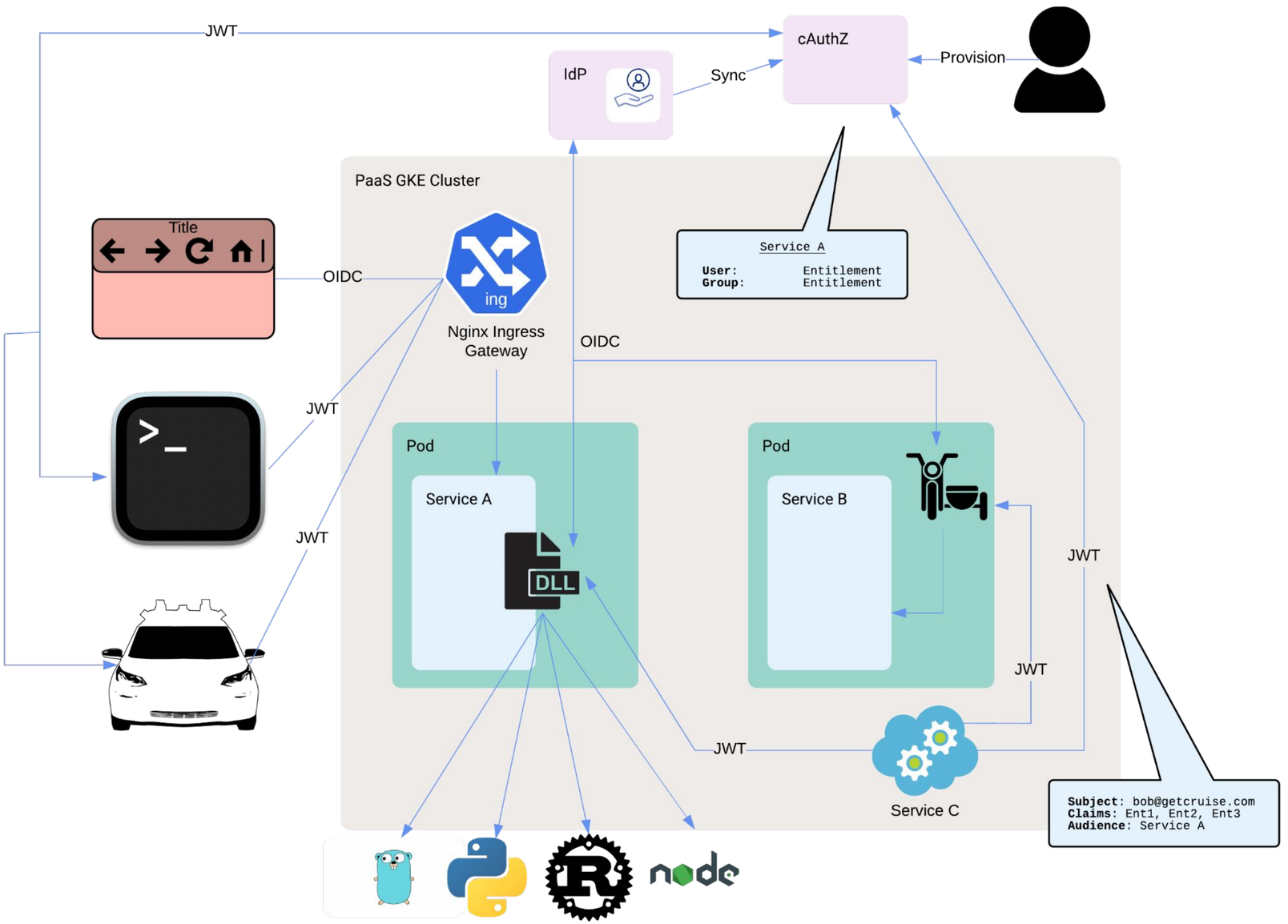


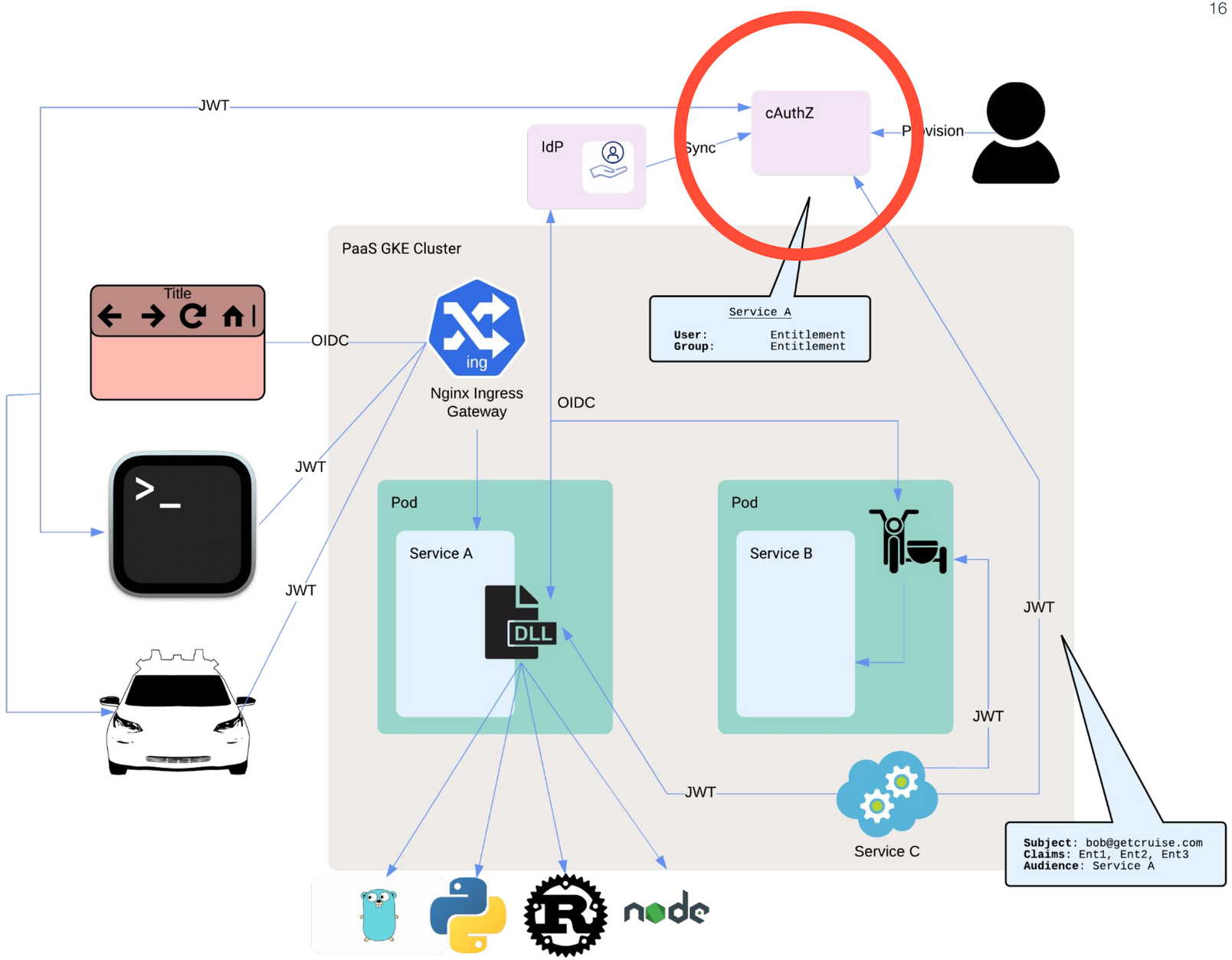
- Choose an authentication and authorization standard:
  - **AuthN**: OIDC with a single IdP (for humans)
  - **AuthZ**: A custom, entitlement-based authorization broker called **cAuthZ**

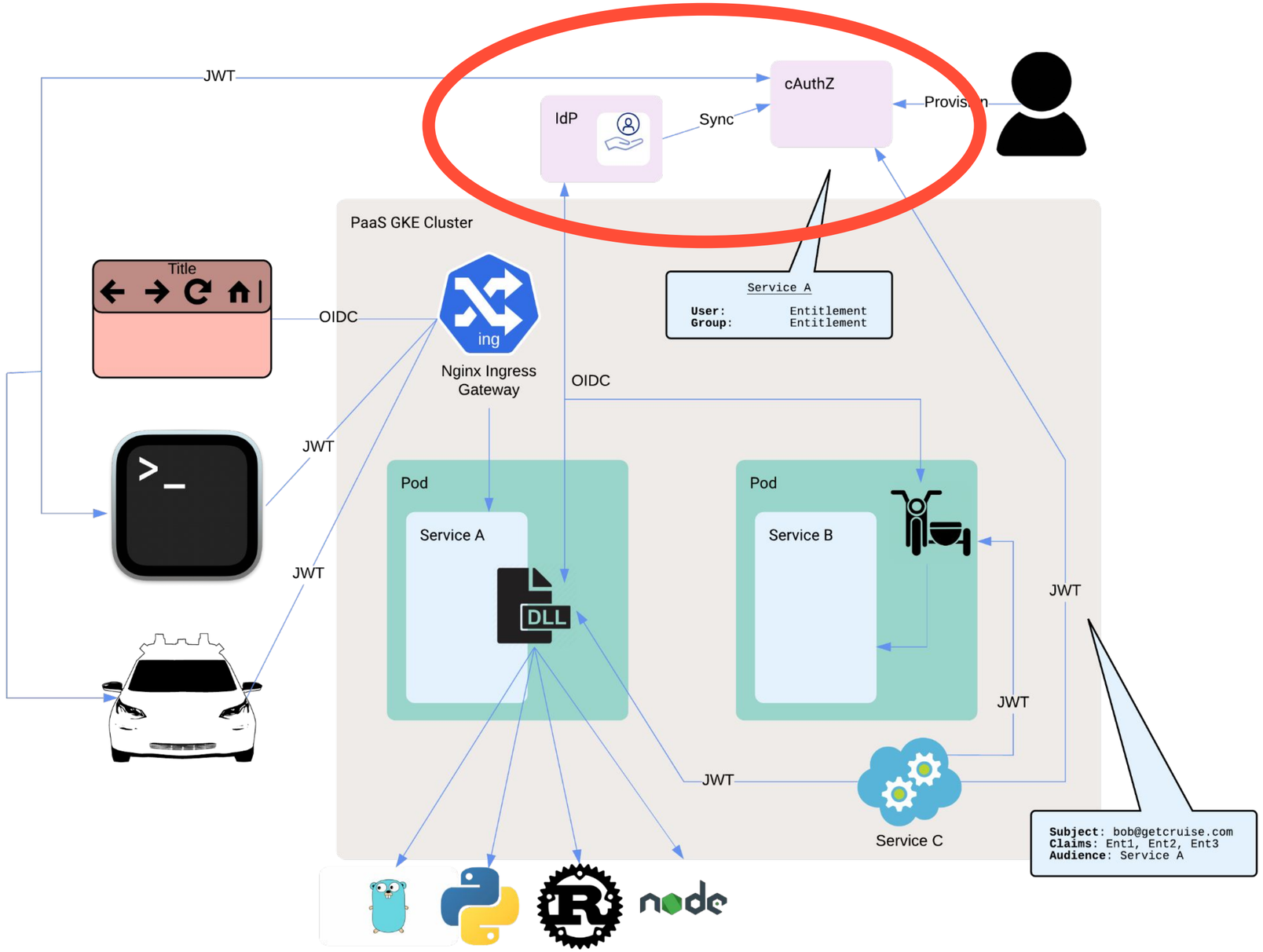
- Build **self-service tools** to promote use of the standard:
  - Identity Orchestration Platform (**IOP**)

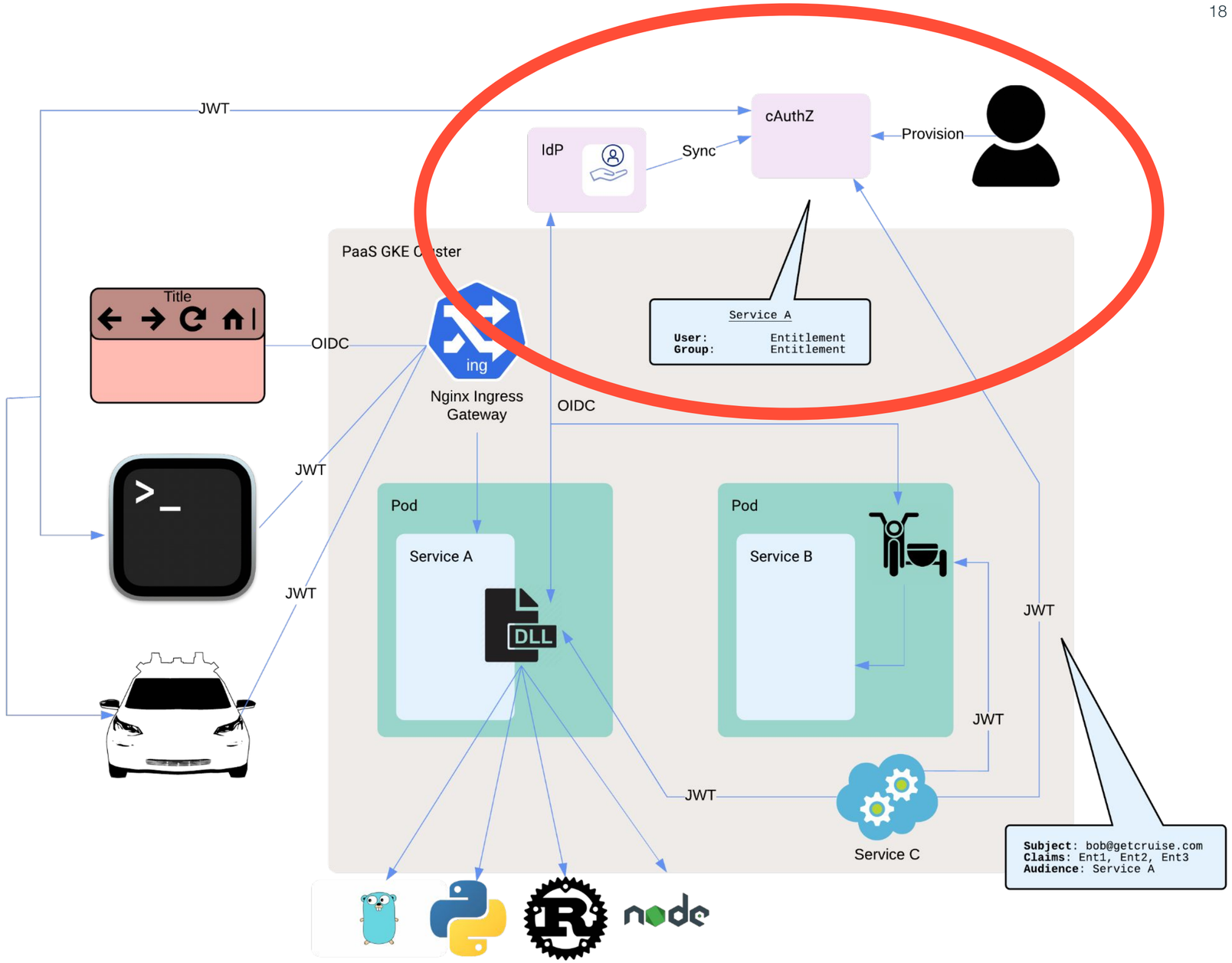
- Build frameworks and libraries to foster implementation:
  - **Opinionated** AuthN/Z libraries in commonly used languages at Cruise
  - Cruise Auth Proxy (**CAP**) sidecar
  - **AuthCLI**: A CLI tool used for retrieving and storing non-browser client credentials, (for humans)

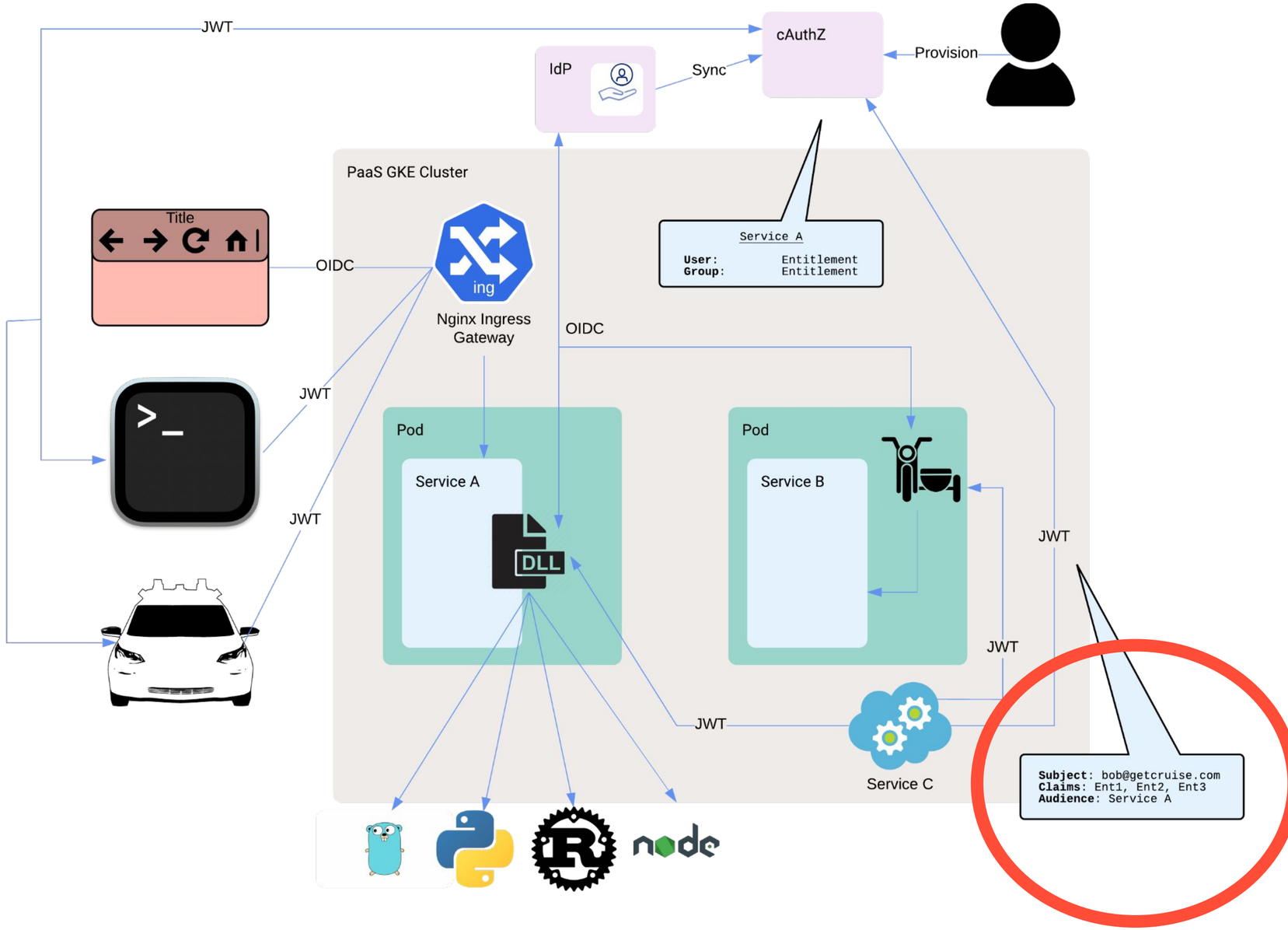
- Standardize on offline tokens (JWTs) minted by **cAuthZ**.

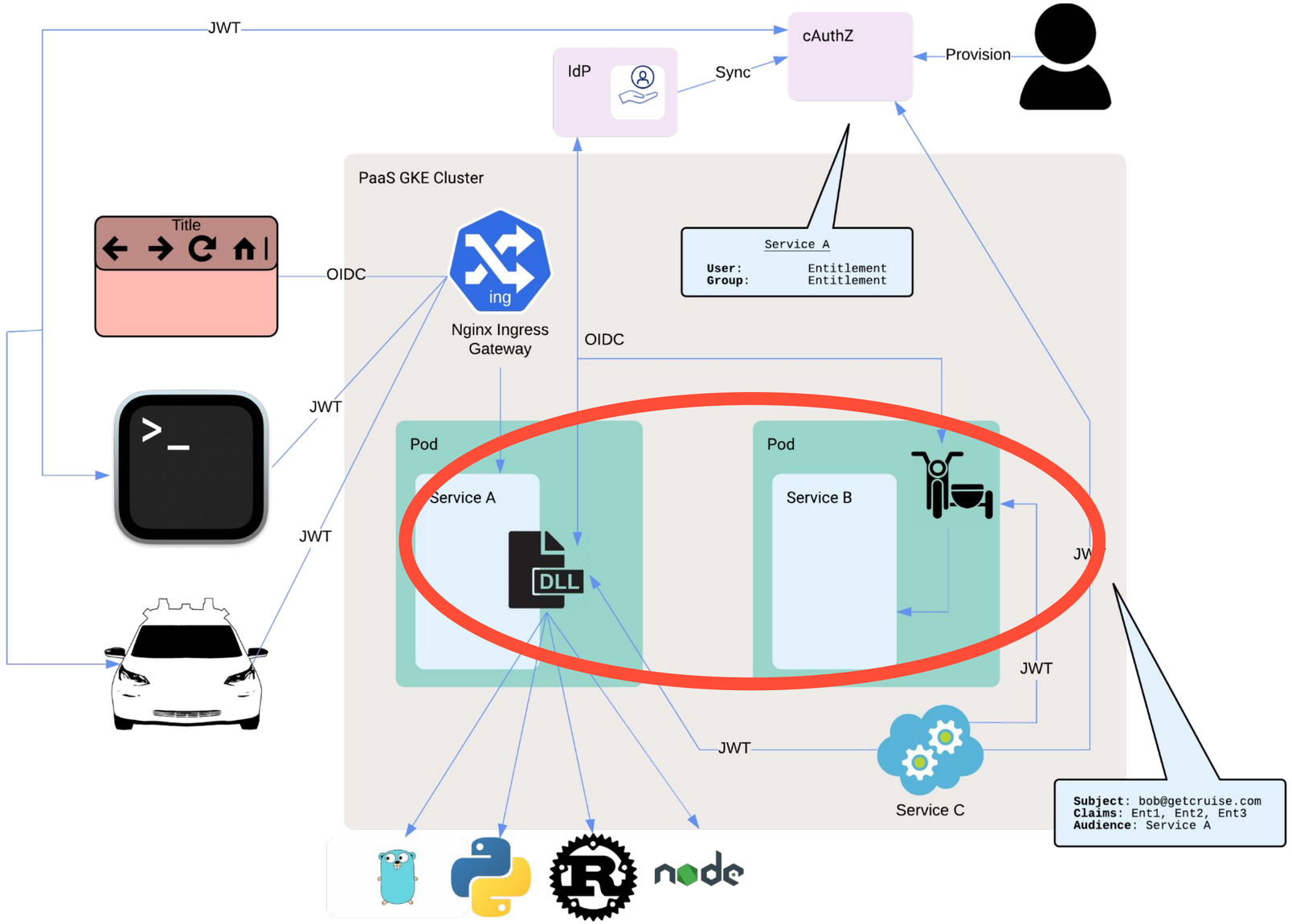

JWT
cAuthZ
Provision
IdP
Sync
PaaS GKE Cluster
Title
OIDC
ing
Nginx Ingress Gateway
Service A
User: Entitlement
Group: Entitlement
OIDC
JWT
Pod
Service A
DLL
Pod
Service B
JWT
JWT
JWT
JWT
Service C
Subject: bob@getcruise.com
Claims: Ent1, Ent2, Ent3
Audience: Service A


JWT
cAuthZ
IdP
Sync
Provision
PaaS GKE Cluster
Title
OIDC
ing
Nginx Ingress Gateway
Service A
User: Entitlement
Group: Entitlement
OIDC
JWT
JWT
Pod
Service A
DLL
Pod
Service B
JWT
JWT
JWT
Service C
Subject: bob@getcruise.com
Claims: Ent1, Ent2, Ent3
Audience: Service A


JWT
cAuthZ
IdP
Sync
Provisi n
PaaS GKE Cluster
Title
OIDC
ing
Nginx Ingress Gateway
Service A
User: Entitlement
Group: Entitlement
OIDC
JWT
JWT
Pod
Service A
DLL
Pod
Service B
JWT
JWT
JWT
Service C
Subject: bob@getcruise.com
Claims: Ent1, Ent2, Ent3
Audience: Service A




JWT
cAuthZ
Provision
IdP
Sync
PaaS GKE Cluster
Title
OIDC
ing
Nginx Ingress Gateway
Service A
User: Entitlement
Group: Entitlement
OIDC
JWT
JWT
Pod
Service A
DLL
Pod
Service B
JWT
JWT
JWT
Service C
Subject: bob@getcruise.com
Claims: Ent1, Ent2, Ent3
Audience: Service A


JWT
cAuthZ
Provision
IdP
Sync
PaaS GKE Cluster
Title
OIDC
ing
Nginx Ingress Gateway
Service A
User: Entitlement
Group: Entitlement
OIDC
JWT
JWT
Pod
Service A
DLL
Pod
Service B
JWT
JWT
Service C
Subject: bob@getcruise.com
Claims: Ent1, Ent2, Ent3
Audience: Service A

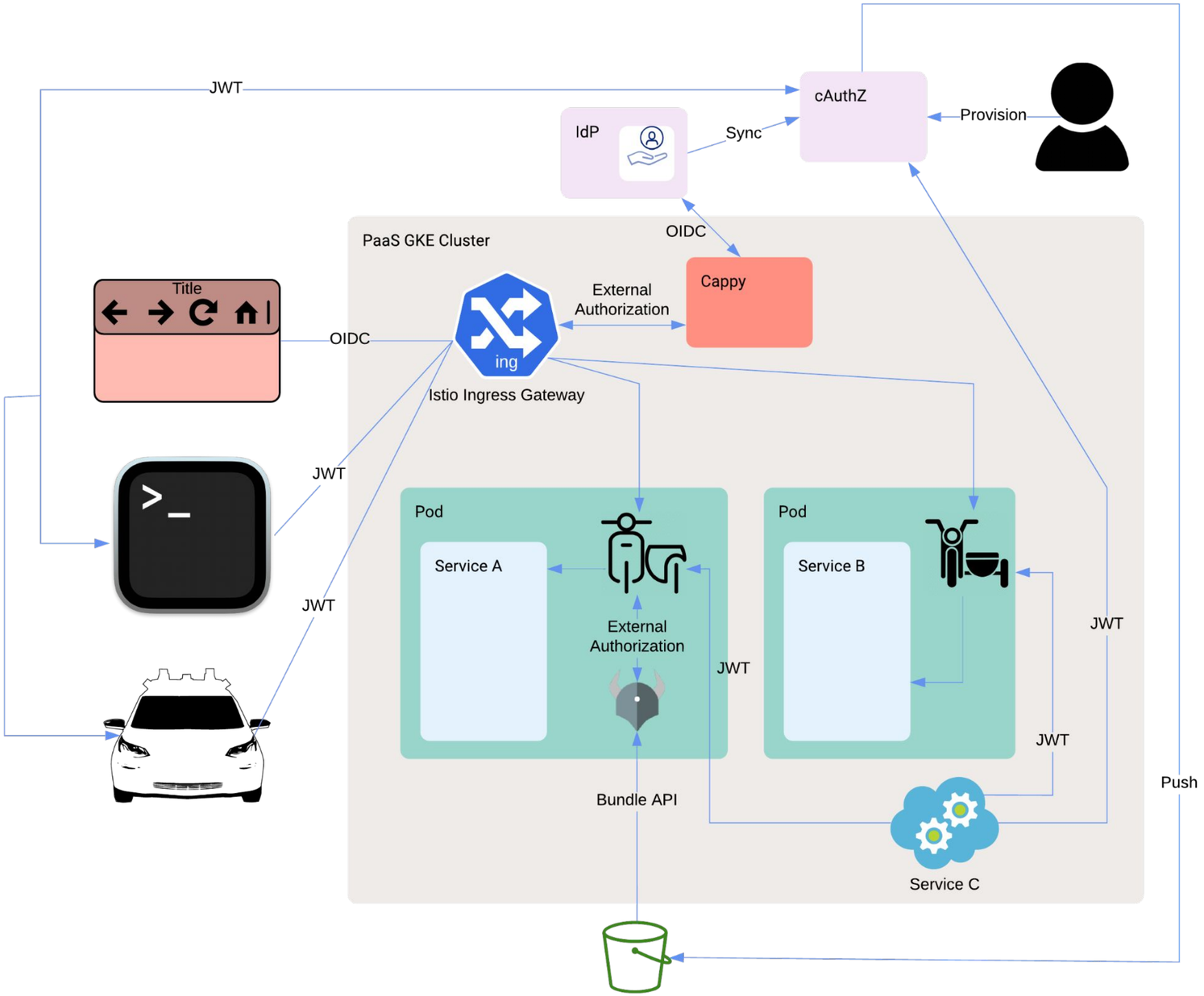
- **Problems we have at the walk stage:**
  - **No enforcement**
  - **Lots of work required by service owners to implement AuthN/Z**
    - Code changes required
  - **Token size is exploding**
  - **Maintaining auth libraries is hard**
    - Many languages to support
    - Indeterminate update cycle
  - **Token process & policies all live in the depths of each service**
    - Not easy to audit
    - Can’t iterate to solve problems

Make AuthN/Z Better

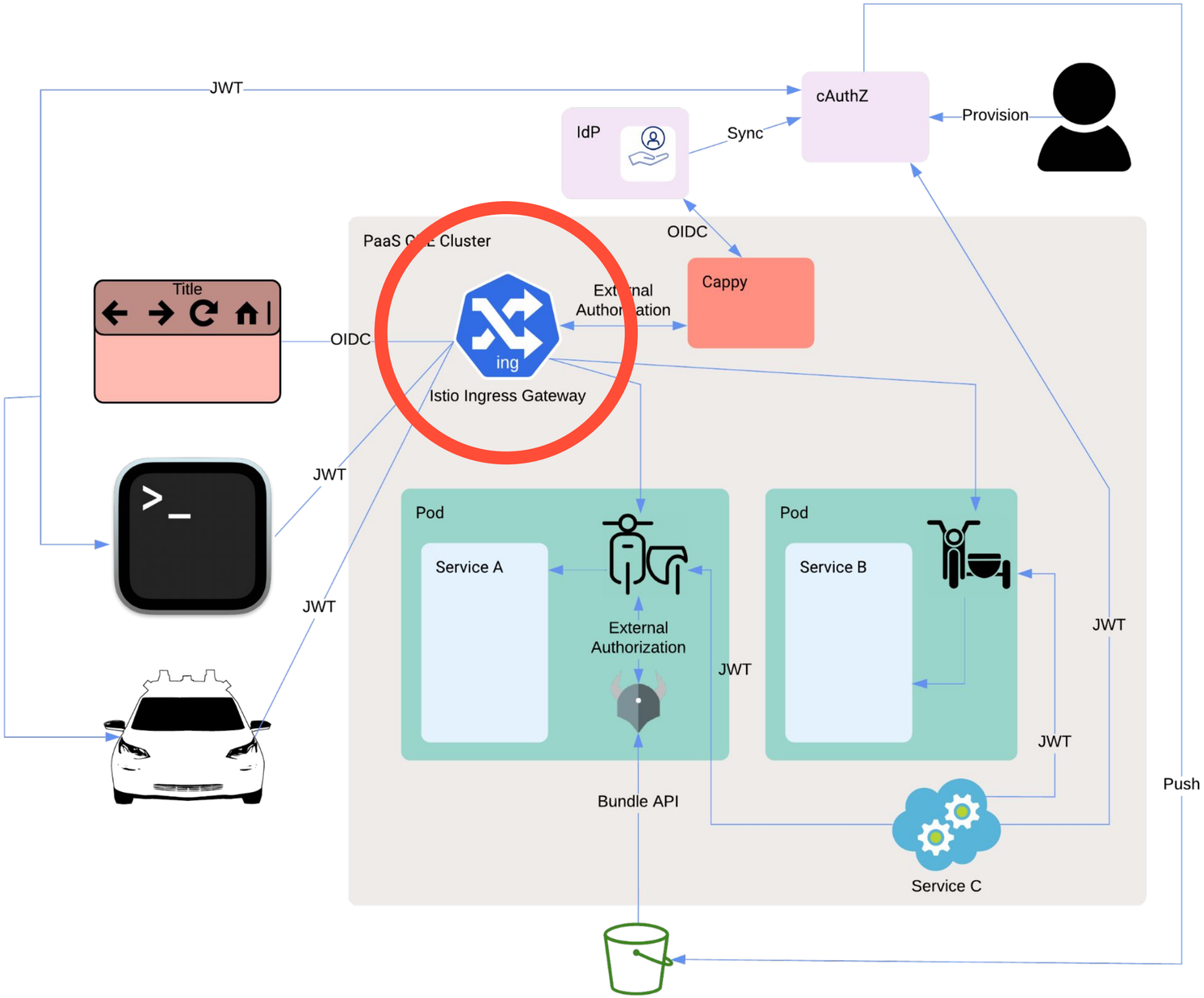
Phase III: ~~Run~~ Speed Walk

Open Policy Agent

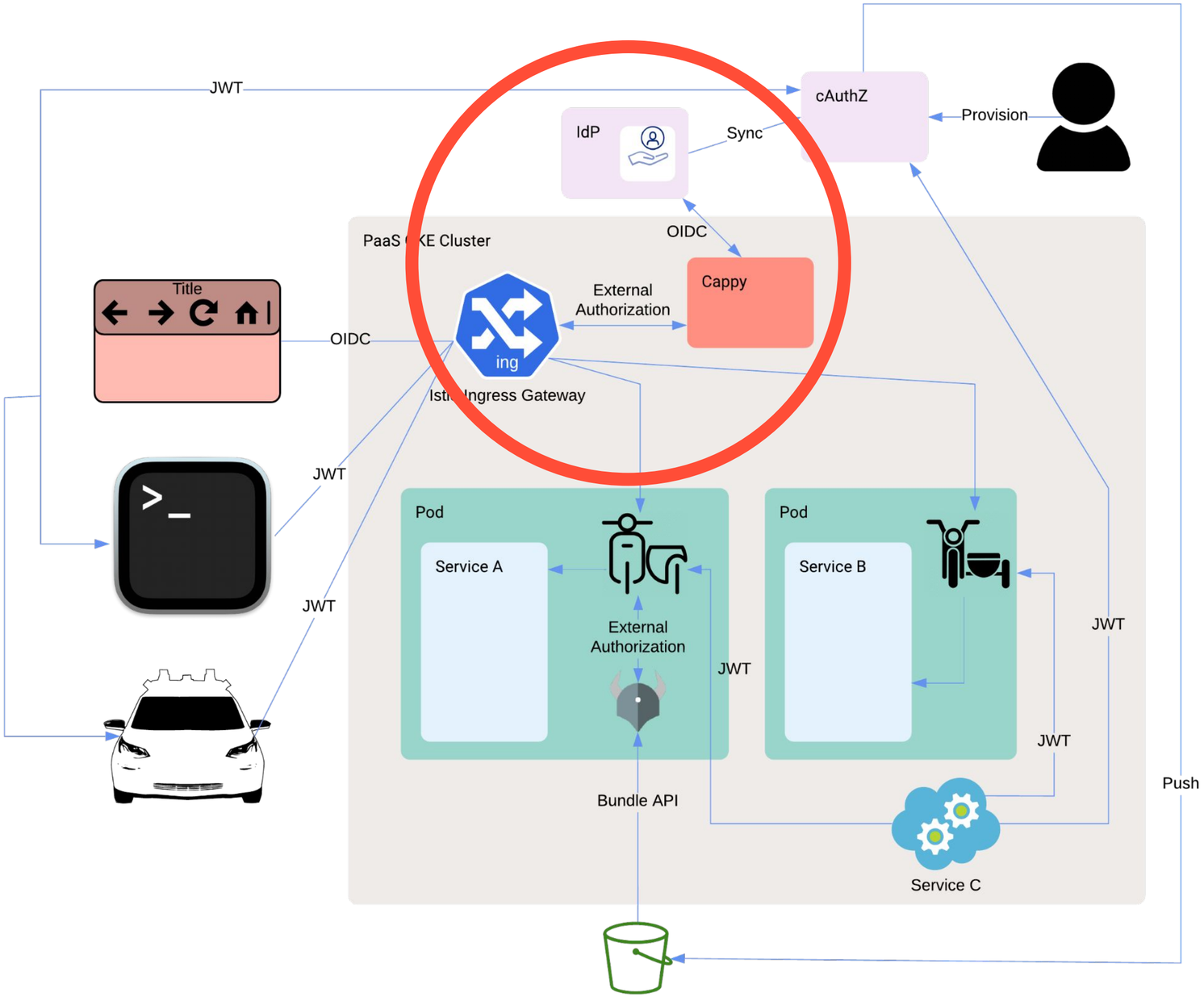
- Make authentication and authorization **opt-out instead of opt-in**
- Standardize on industry-supported **open-source** software over our custom solutions
- Move the service policy **out of the depths of service code**
- Do this **without** requiring any (or very little) changes to services



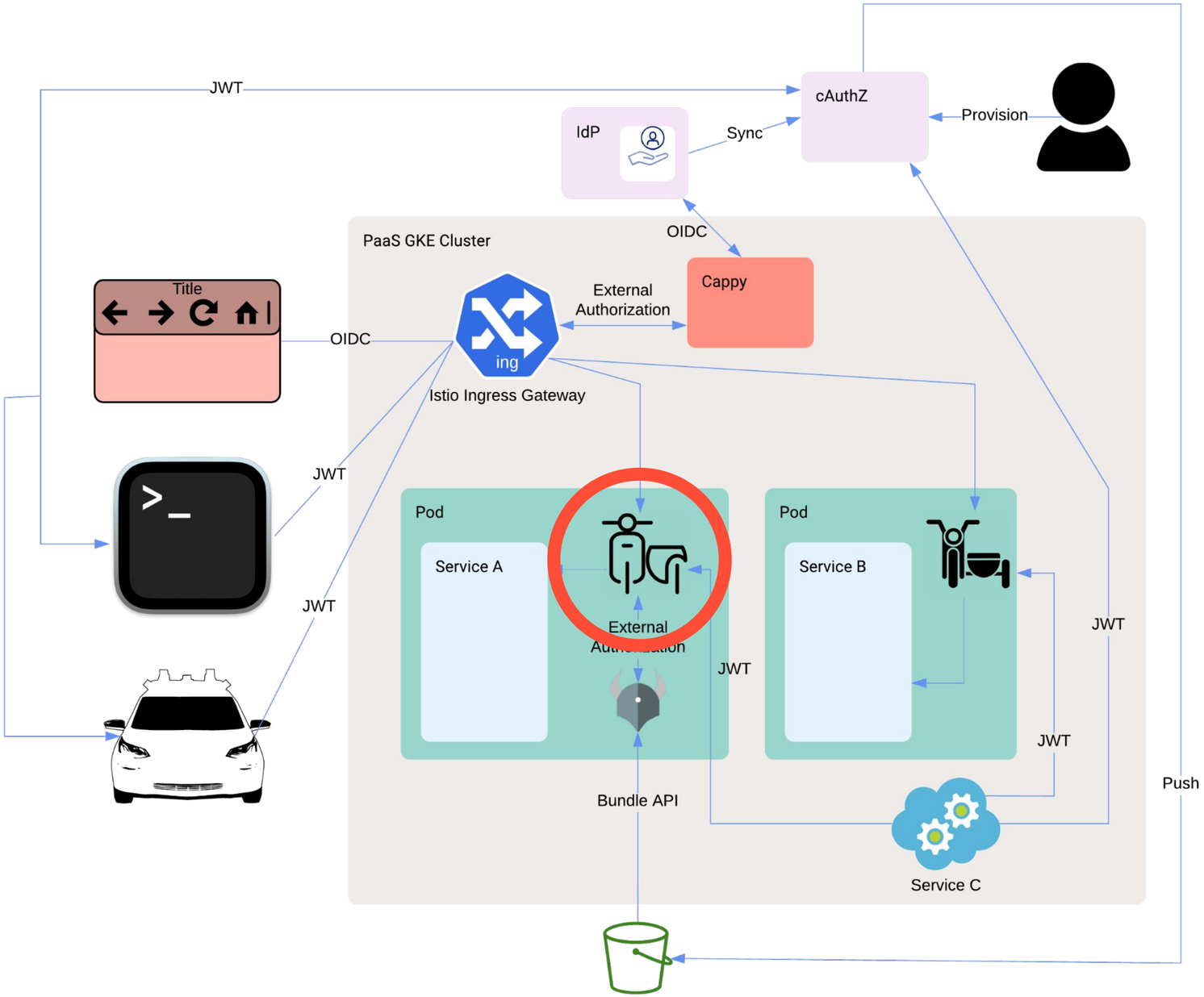
Istio Ingress



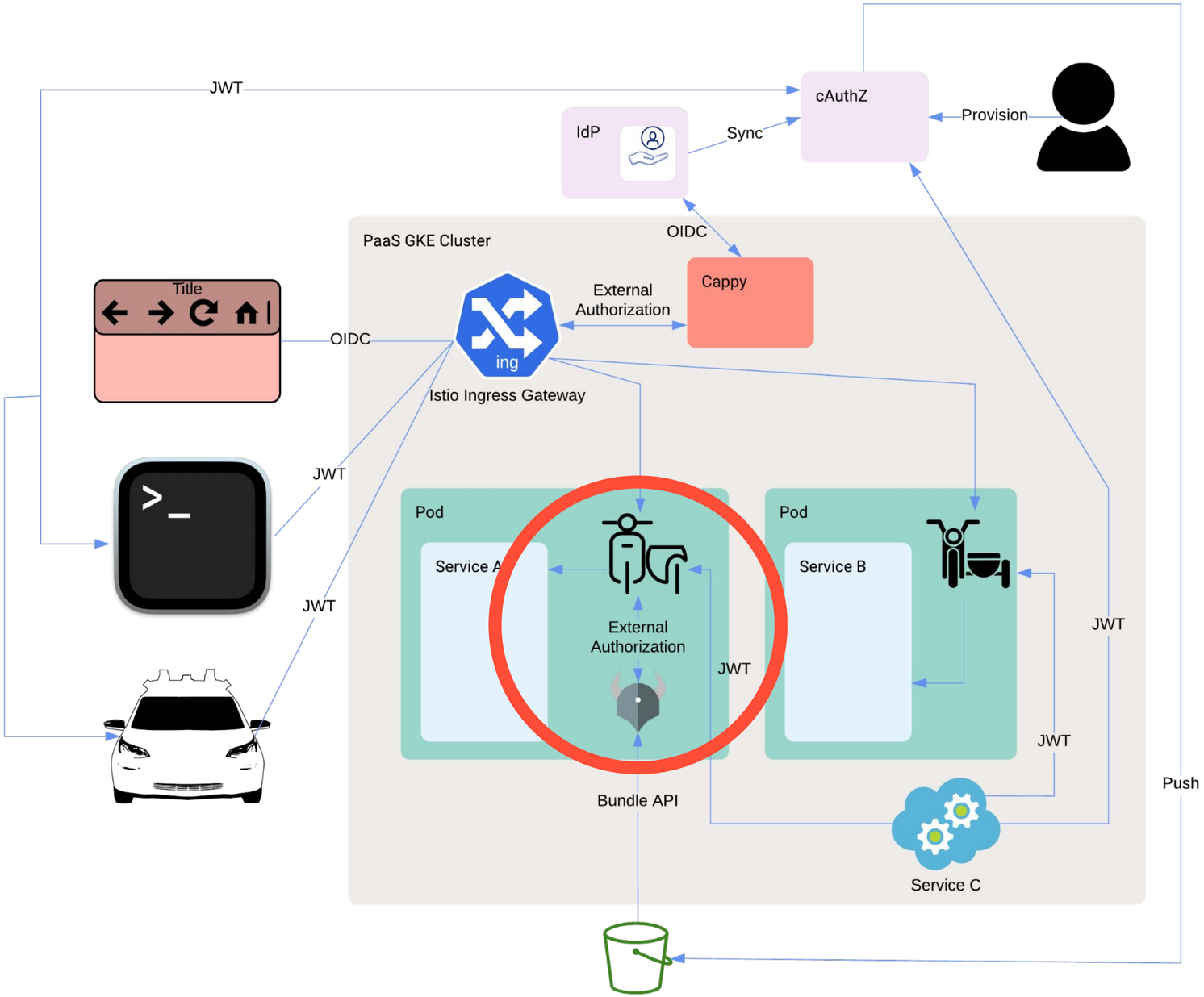
Istio Ingress



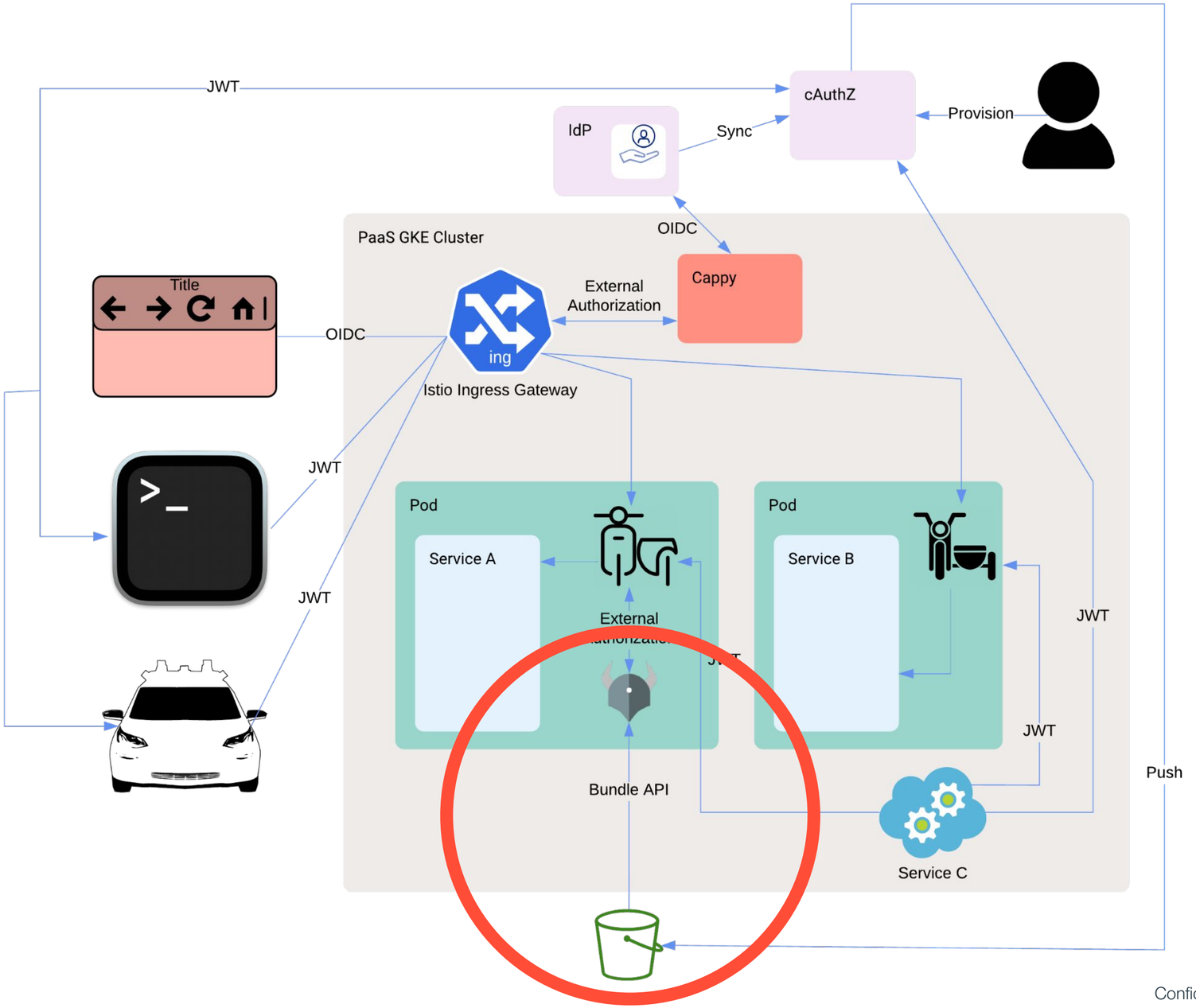
Istio Ingress
Istio Proxy

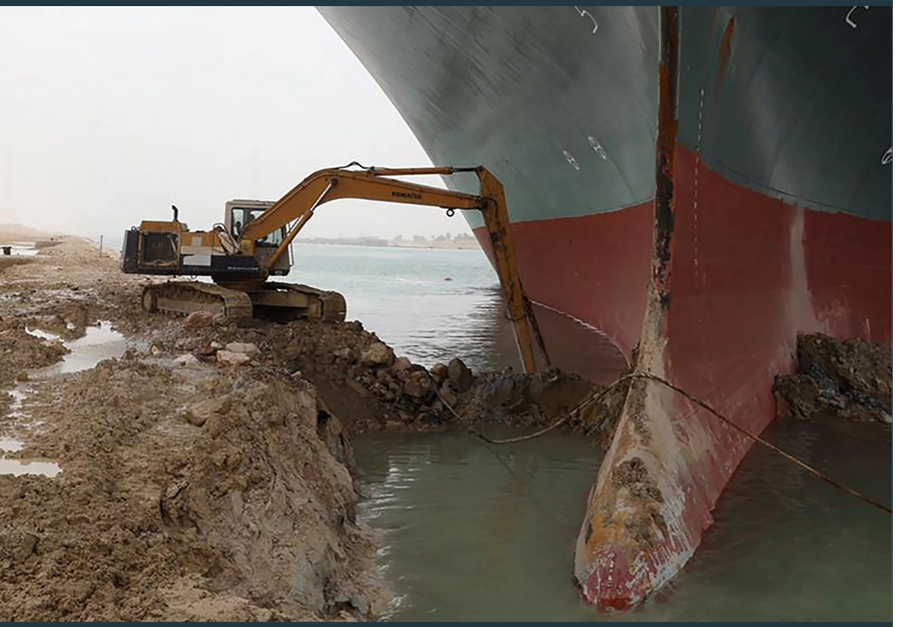


Istio Ingress
Istio Proxy
Open Policy Agent



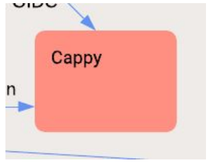
Istio Ingress
Istio Proxy
Open Policy Agent





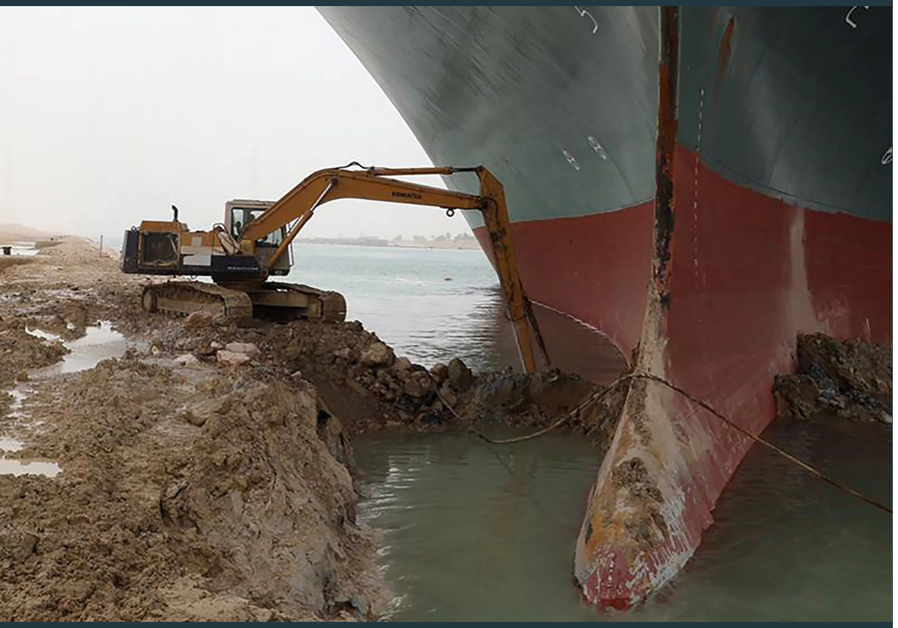
- **Nginx -> Envoy Ingress**
  - Ingress -> VirtualService
  - `istioctl convert-ingress`
  - SxS via DNS CNAME

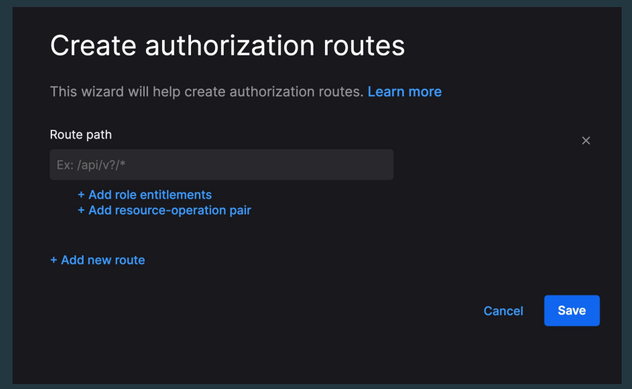
- **Onboarding to Cappy**
  - Start with opt-in
  - Double authentication requires no service changes
  - Integrate with existing libraries & sidecars



- **Istio Proxy**
  - `istio-injection=enabled`

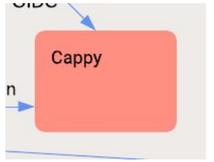
- **OPA Agent**
  - Start with opt-in
  - `opa-agent=enabled`
  - Single rego with different data per-service
  - Coarse-grained, route-based authorization (stateless)
  - Fine-grained authorization is still done in service (stateful)





- **Nginx -> Envoy Ingress**
  - Ingress -> VirtualService
  - `istioctl convert-ingress`
  - SxS via DNS CNAME

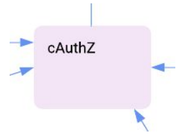
- **Onboarding to Cappy**
  - Start with opt-in
  - Double authentication requires no service changes
  - Integrate with existing libraries & sidecars



- **Istio Proxy**
  - `istio-injection=enabled`

- **OPA Agent**
  - Start with opt-in
  - `opa-agent=enabled`
  - Single rego with different data per-service
  - Coarse-grained, route-based authorization (stateless)
  - Fine-grained authorization is still done in service (stateful)

- **Policy Generation and Deployment**
  - New UI & CI/CD process to push policies to cAuthZ
  - Serialization of policies & group membership pushed to buckets



- **Authentication and authorization are now opt-out and not opt-in**
  - “all our services have authN & authZ”
- **Authorization policy lives outside of the service**
  - Managed, iterated on & audited centrally
- **Using open-source and battle tested software instead of custom solutions**
  - Participate in open-source ecosystem where our contributions help others
- **Can iterate without having to effect service owners**
  - Create new auth paradigms without requiring work be done on each service

- Istio sidecars can be leveraged further
  - Transparent mTLS between services (inc. ingress -> service)
  - SPIFFE service identity (to replace API keys)

- Support custom rego policies for both stateful and stateless authorization
  - Rego modules can be (re)used to provide vetted functionality
  - CI pipeline around rego can provide lint, validation, approval, etc.

cruise

We’re hiring!

getcruise.com/careers

Nitish Krishna
Poompatnam
@NitishKrishna1

Roman Porter
@r0m1es

James Barclay
@futureimperfect

Thank you!