AWS re:Inforce

JUNE 13 - 14, 2023 | ANAHEIM, CA

IAM357

Multi-tenant SaaS identity with Amazon Cognito user pools

Tony Fendall (he/him)
Principal Solutions Architect
AWS

Andrew Hooker (he/him)
Enterprise Solutions Architect
AWS

Suresh Sridharan (he/him)
Sr. Product Manager – Amazon Cognito
AWS

Abrom Douglas (he/him)
Solutions Architect – Amazon Cognito
AWS

What we will cover today

Amazon Cognito and multi-tenancy

Hands-on lab

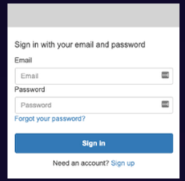
Great discussion

Amazon Cognito – Introduction

User pools

Directory for your identities

- User directory for web/mobile app
- Issue tokens
- OAuth2
- Client credentials



Sign in with your email and password

Email

Email

Password

Password

Forgot your password?

Need an account? Sign up

Hosted UI

Issues

JSON
Web Tokens (JWTs)

Federation

Identity pools

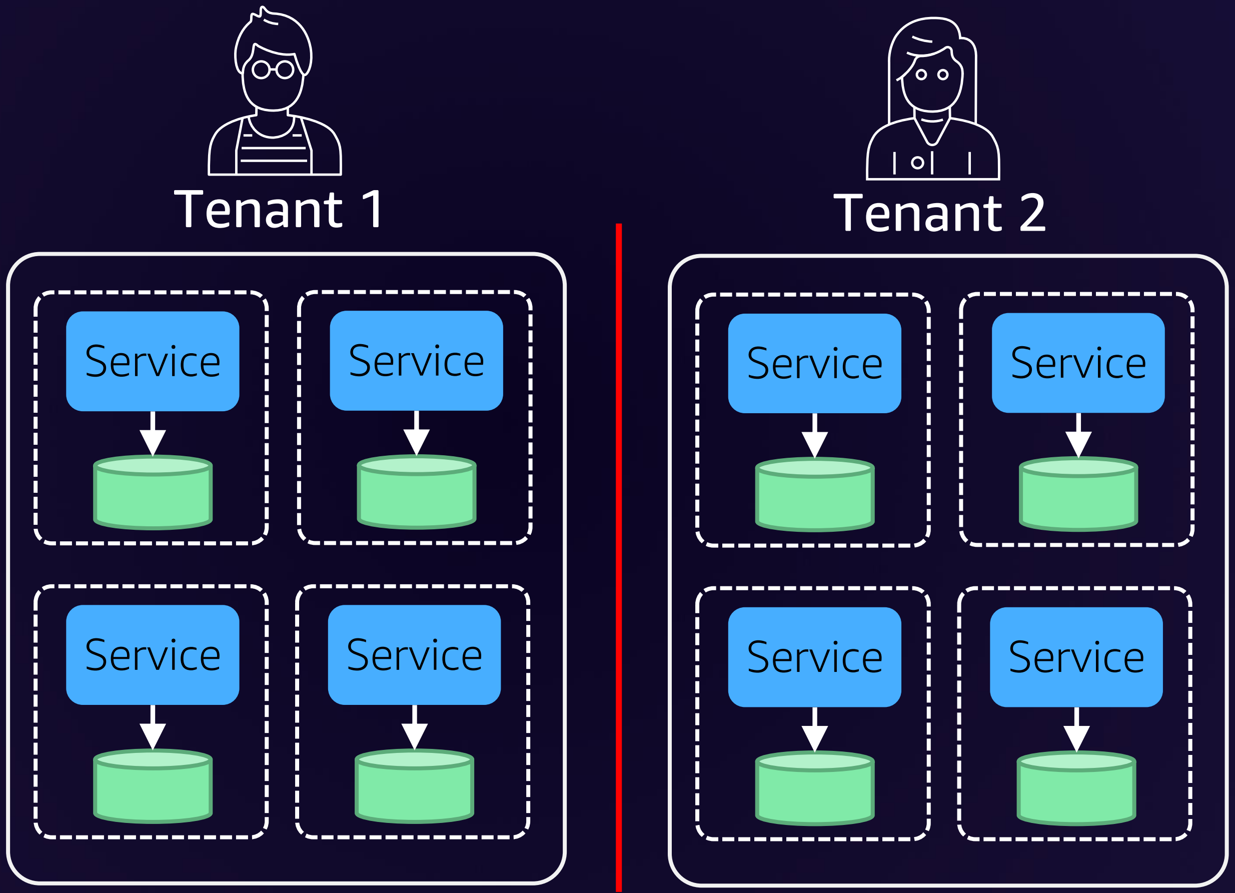
Access control for AWS resources

- Federated identities
- Credentials broker
- Guest access
- Authenticated access
- Fine-grained access

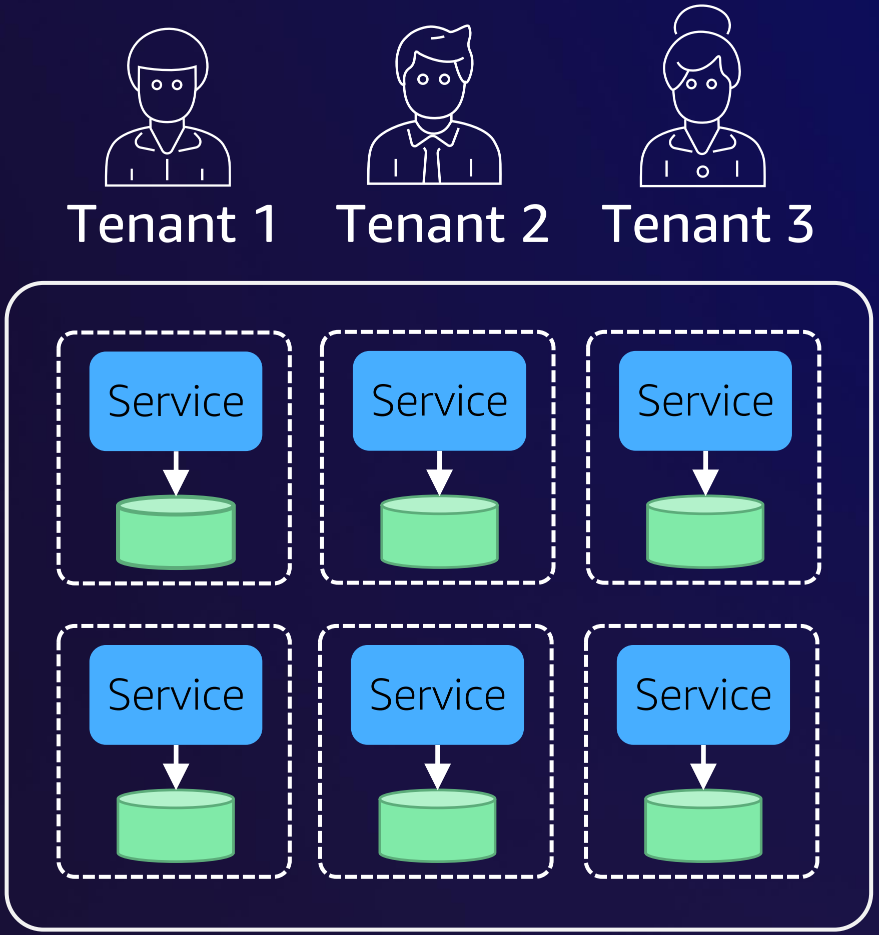
Provides

AWS temporary
credentials

Multiple flavors of isolation



Isolation through siloed infrastructure
(Silo model)

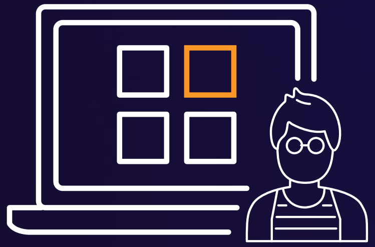


Isolation through runtime policies
(Pool model)

Isolation flavors in Amazon Cognito

Silo isolation

By user pool



By app client

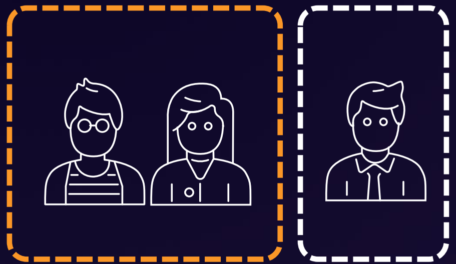
Pros

- Coarse-grained isolation
- Customer acceptance
- Better tool alignment

Cons

- Deployment
- Cost optimization
- Manageability

Policy isolation



By user group

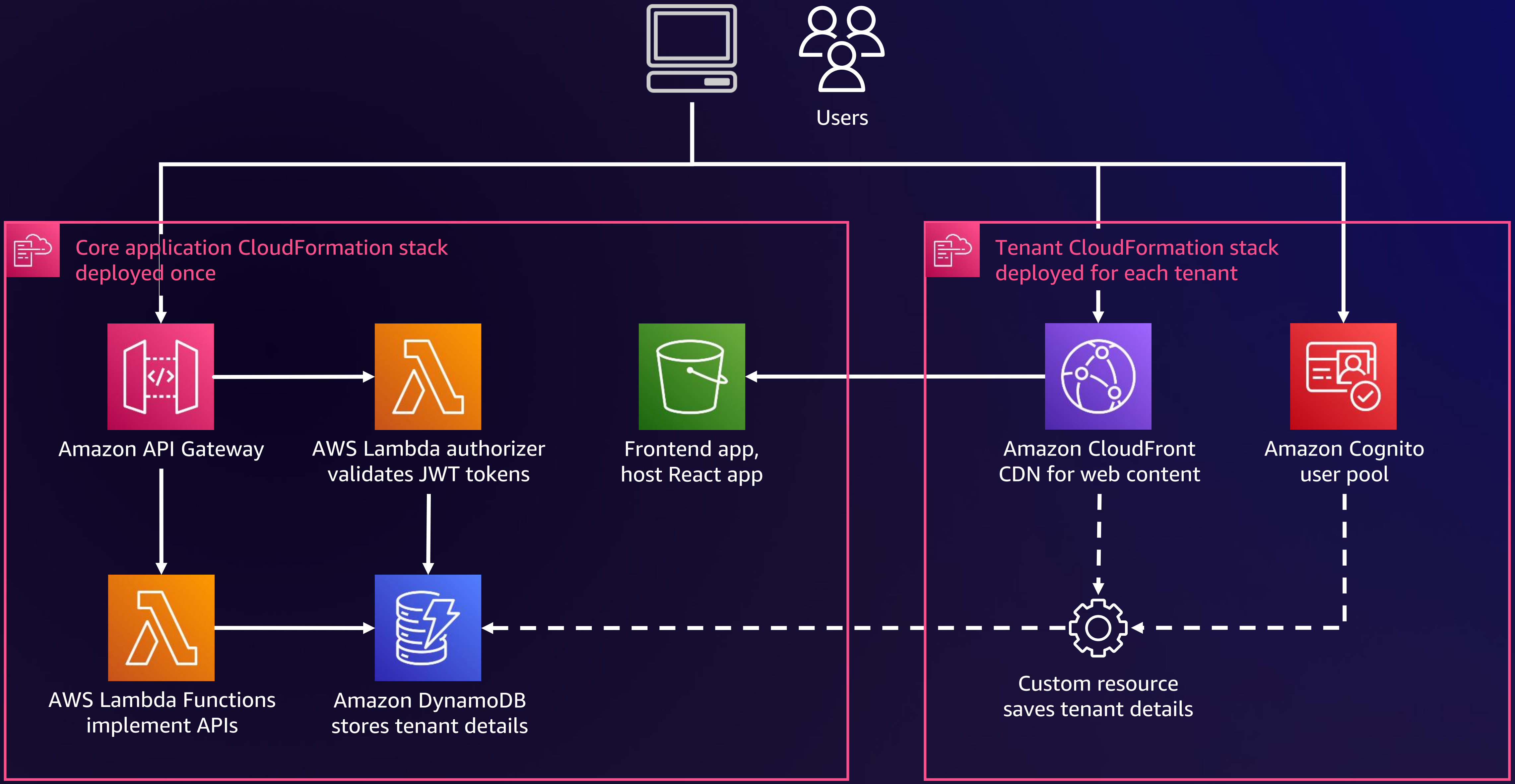
By user attribute

Pros

- Fine-grained isolation
- Enables resource pooling
- Flexibility

Cons

- Customer acceptance
- Relies on convention
- Mix of technologies


Users
Core application CloudFormation stack
deployed once
Amazon API Gateway
AWS Lambda authorizer
validates JWT tokens
Frontend app,
host React app
AWS Lambda Functions
implement APIs
Amazon DynamoDB
stores tenant details
Tenant CloudFormation stack
deployed for each tenant
Amazon CloudFront
CDN for web content
Amazon Cognito
user pool
Custom resource
saves tenant details

Accessing the workshop

https://bit.ly/reInforce23-IAM357

Multi-tenancy security recommendations

- Use only a **verified email address to authorize** user access to a tenant based on **domain match**
- Use ***immutable*** **or read-only attributes** for the user profile attributes that **identify tenants**
- Use **1:1 mapping between external IdP and application client** to prevent unauthorized cross-tenant access
- If you implement tenant-matching logic in your application, **restrict users so that they can't modify the criteria** that authorizes user access to tenants

https://go.aws/3P12GgV

Thank you!

Please complete
the session survey
in the mobile app

Isolation by user pool

When to use user-pool-based multi-tenancy

- Your application has **different configurations for each tenant**
- Your application has **complex user-to-tenant role mapping**
- Your application uses the **default Amazon Cognito hosted UI** as the primary way for *local* users to authenticate
- Your **application is a silo-per-tenant application**

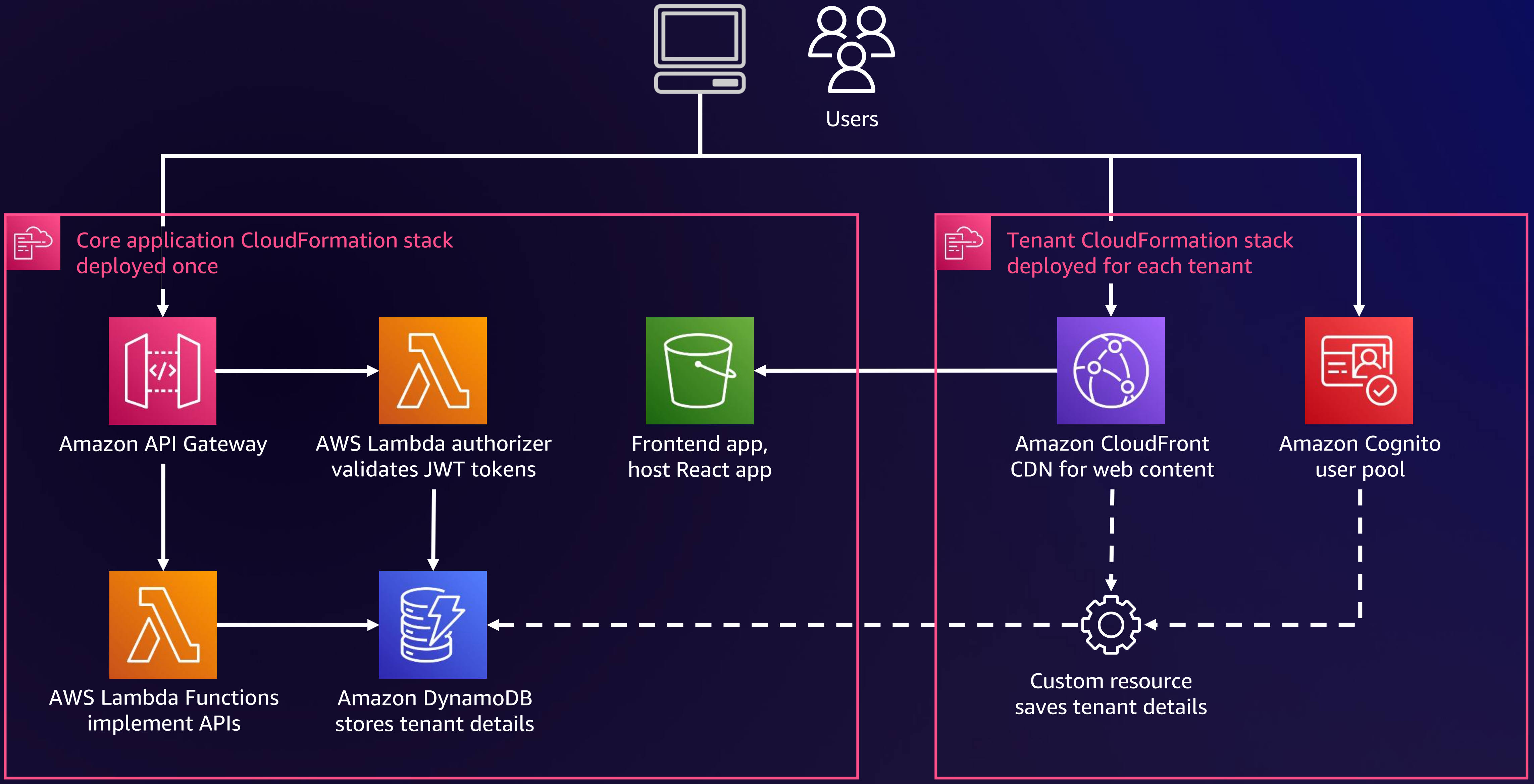
https://go.aws/42ulFDy

Isolation by app client

When to use app-client-based multi-tenancy

- Your application has the **same configuration across all tenants**
- Your application has a **one-to-many mapping between the user and tenants**
- You have a federation-only application where **tenants always use an external IdP to sign in**
- You have a B2B multi-tenant application, and **tenant backend services use a client-credentials grant to access your APIs**

https://go.aws/43Ku8DR



Users

Core application CloudFormation stack deployed once

Amazon API Gateway

AWS Lambda authorizer validates JWT tokens

Frontend app, host React app

AWS Lambda Functions implement APIs

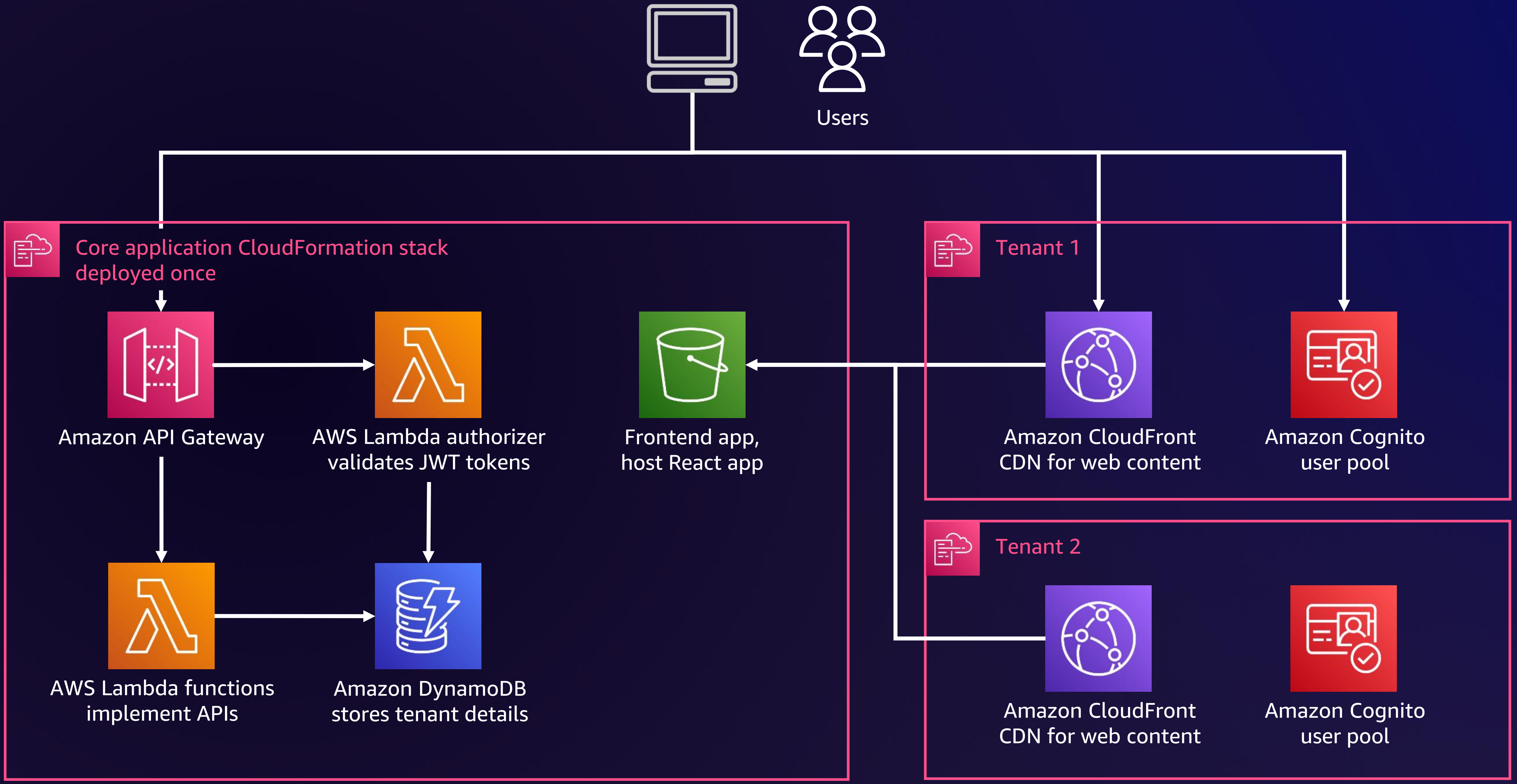
Amazon DynamoDB stores tenant details

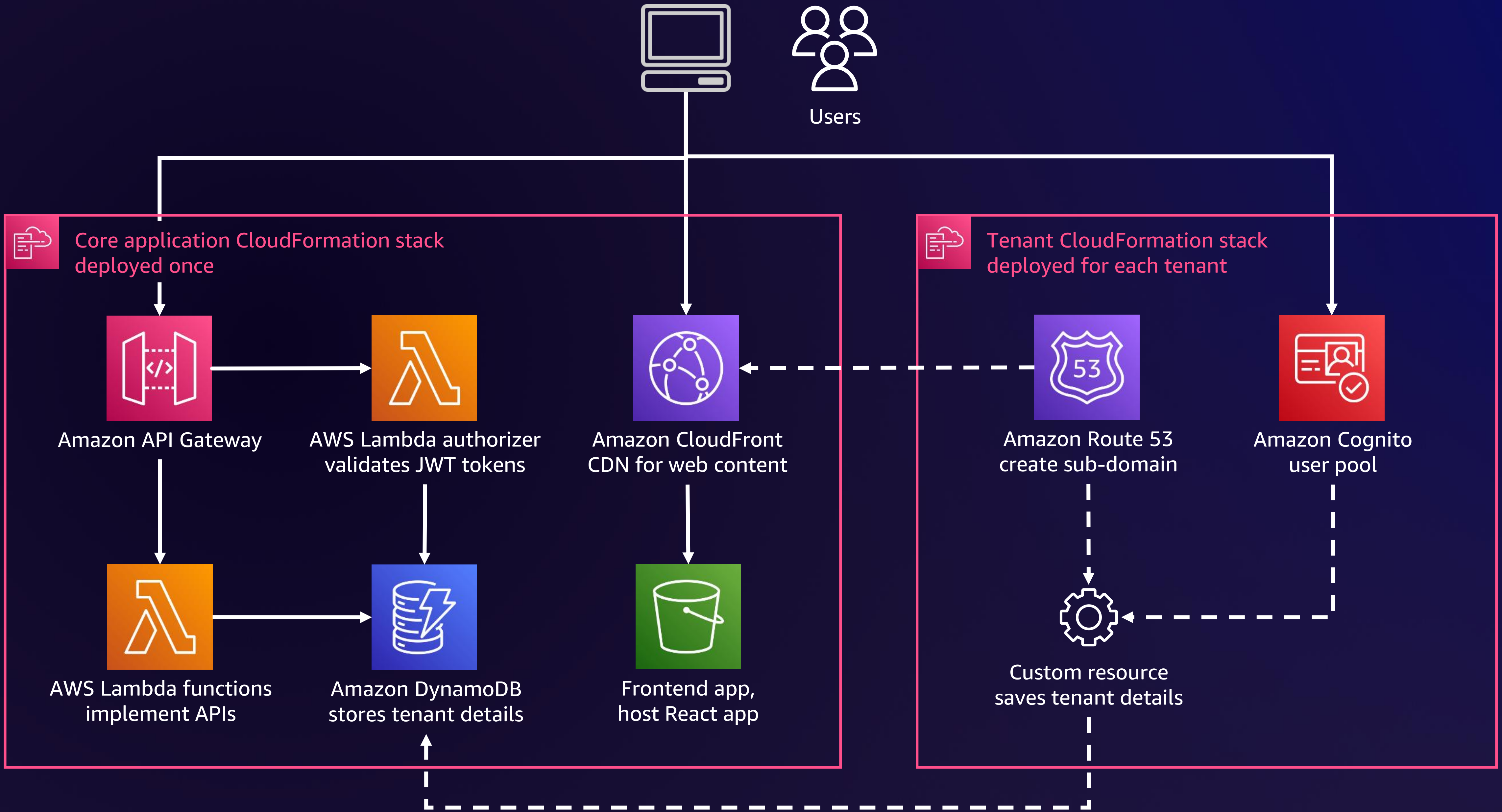
Tenant CloudFormation stack deployed for each tenant

Amazon CloudFront CDN for web content

Amazon Cognito user pool

Custom resource saves tenant details


Users
Core application CloudFormation stack deployed once
Amazon API Gateway
AWS Lambda authorizer validates JWT tokens
Frontend app, host React app
AWS Lambda functions implement APIs
Amazon DynamoDB stores tenant details
Tenant 1
Amazon CloudFront CDN for web content
Amazon Cognito user pool
Tenant 2
Amazon CloudFront CDN for web content
Amazon Cognito user pool


Users
Core application CloudFormation stack deployed once
Amazon API Gateway
AWS Lambda authorizer validates JWT tokens
Amazon CloudFront CDN for web content
AWS Lambda functions implement APIs
Amazon DynamoDB stores tenant details
Frontend app, host React app
Tenant CloudFormation stack deployed for each tenant
53
Amazon Route 53 create sub-domain
Amazon Cognito user pool
Custom resource saves tenant details