

AWS re:Invent

NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV



SAS302

Supporting extensibility in SaaS environments

Bill Tarr

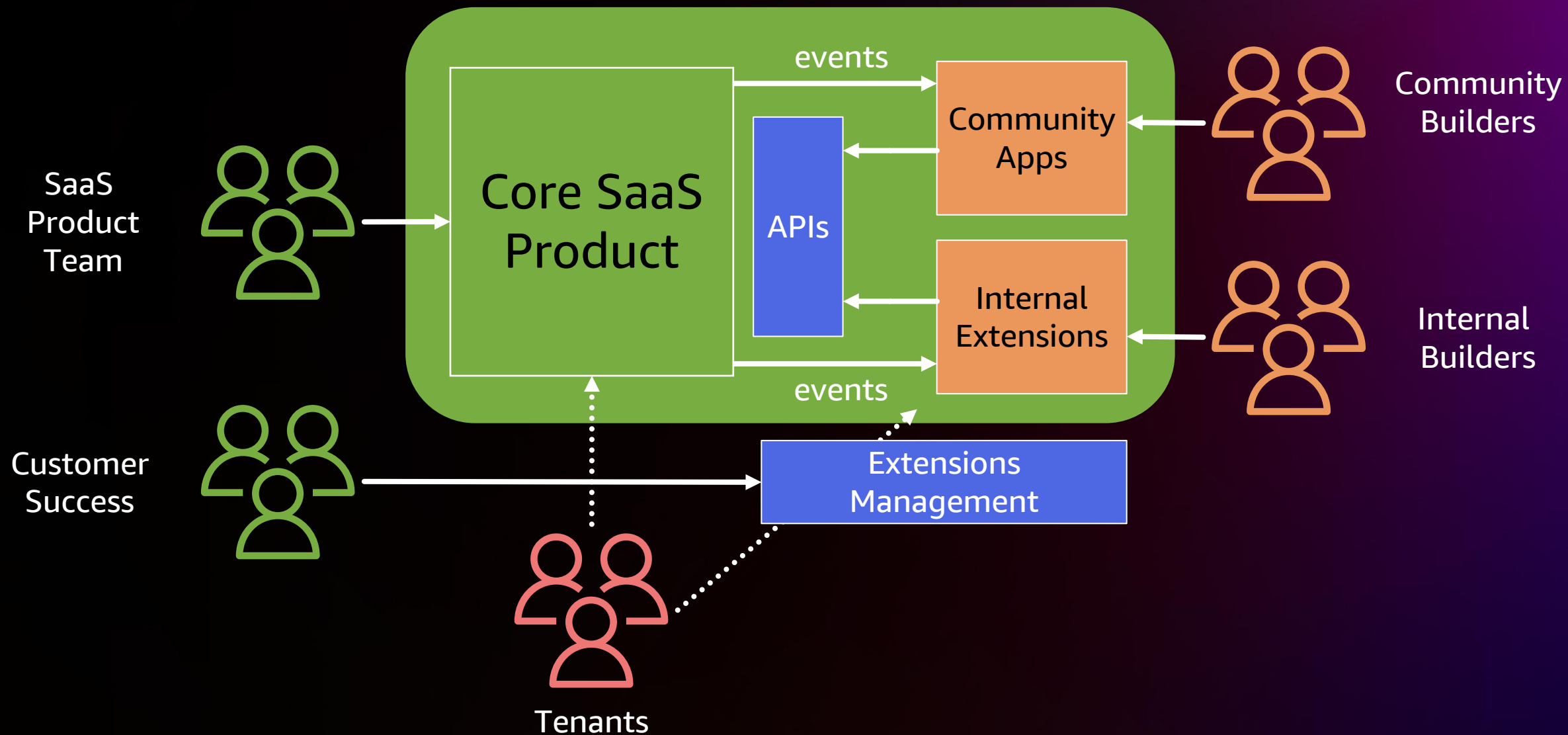
Sr. Partner Solutions Architect
AWS SaaS Factory



What is extensibility in SaaS?

A set of processes allowing a SaaS solution to be extended by developers external to SaaS provider without the need to modify the products core codebase or features.

Why SaaS extensibility matters



Challenges of extensibility



Tenant
Experience



Trusting Code
Contributions

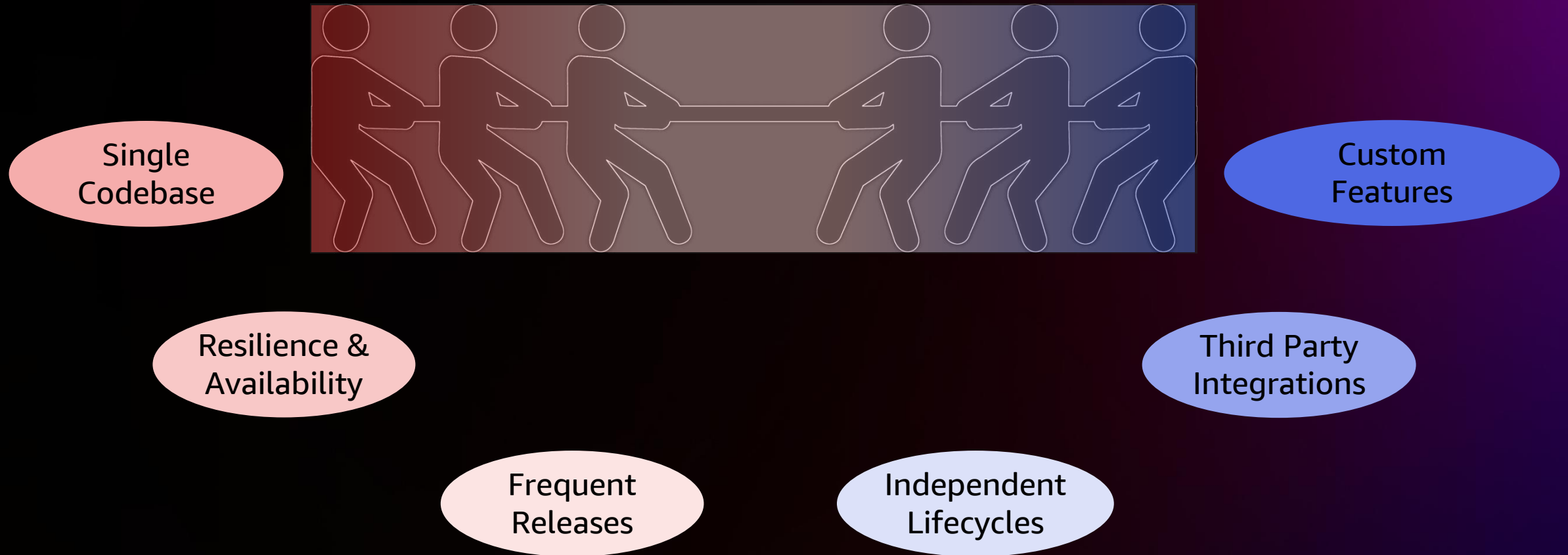


Maintaining
Agility

Maintaining agility with extensibility

Agility

Flexibility



Community Extensibility



Examples of community apps





stripe APPS


Stripe app marketplace example

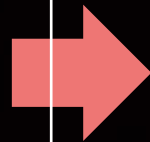
stripe APPS

stripe APP MARKETPLACE

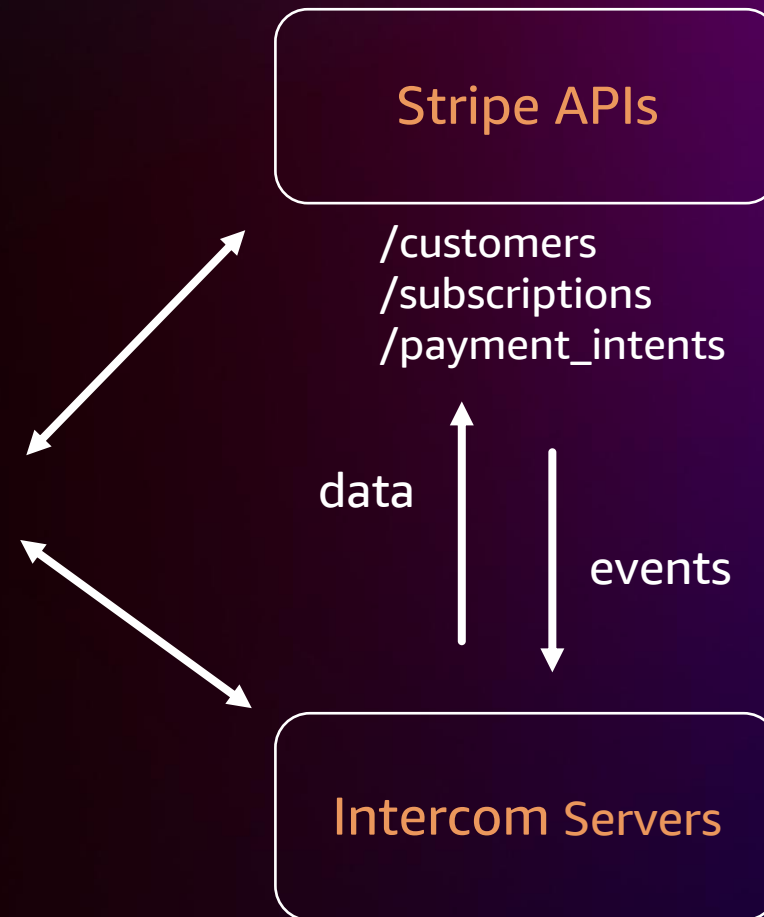
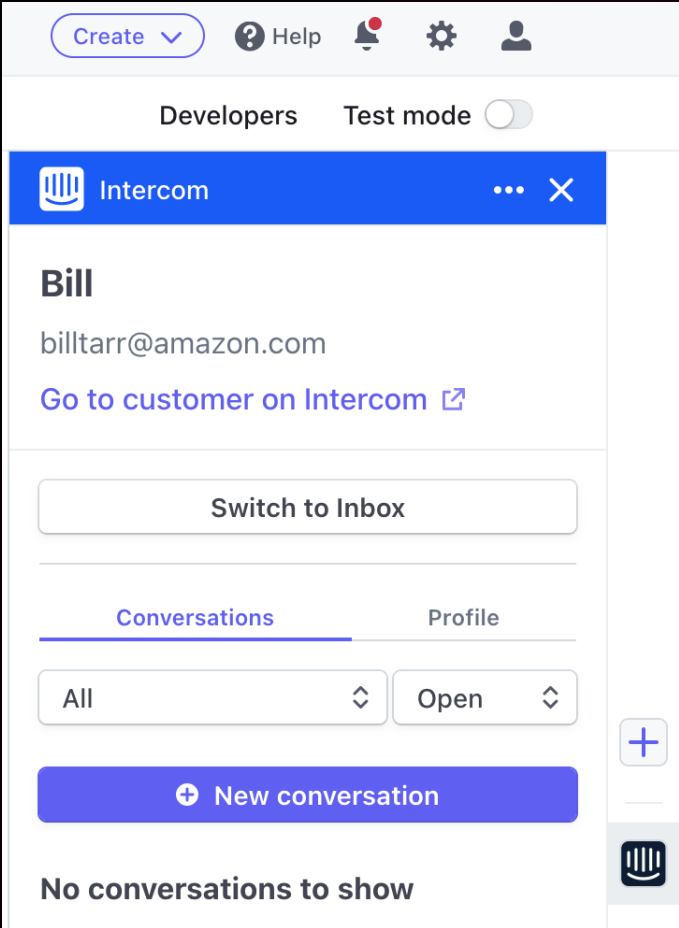
 **Intercom**
See and reply to your Intercom conversations from inside Stripe.

 **Dropbox**
Save files generated in the Stripe Dashboard to Dropbox

 **Capital One**
Manage your small business financial banking needs with Capital One



Intercom app UI



Stripe developer experience

stripe APPS

Granular permissions

Intercom will have access to:

> Charges	Read-only
> PaymentIntents	Read-only
▼ Customers	Read-only

Grants access to Customers and Customer events

This permission also implies the following permission: `billing_clock_read`

Simple SDKs

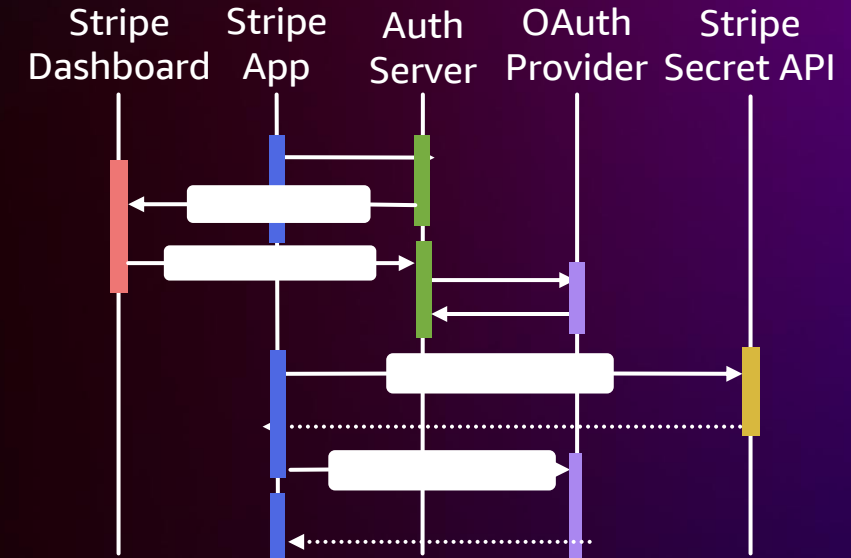
```
const customer = await stripe.customers.create();  
const ephemeralKey = await stripe.ephemeralKeys.create(  
  {customer: customer.id},  
  {apiVersion: '2022-11-15'}  
);
```

for Public APIs

ENDPOINTS

```
POST /v1/customers  
GET /v1/customers/:id  
POST /v1/customers/:id
```

Authorization Flows

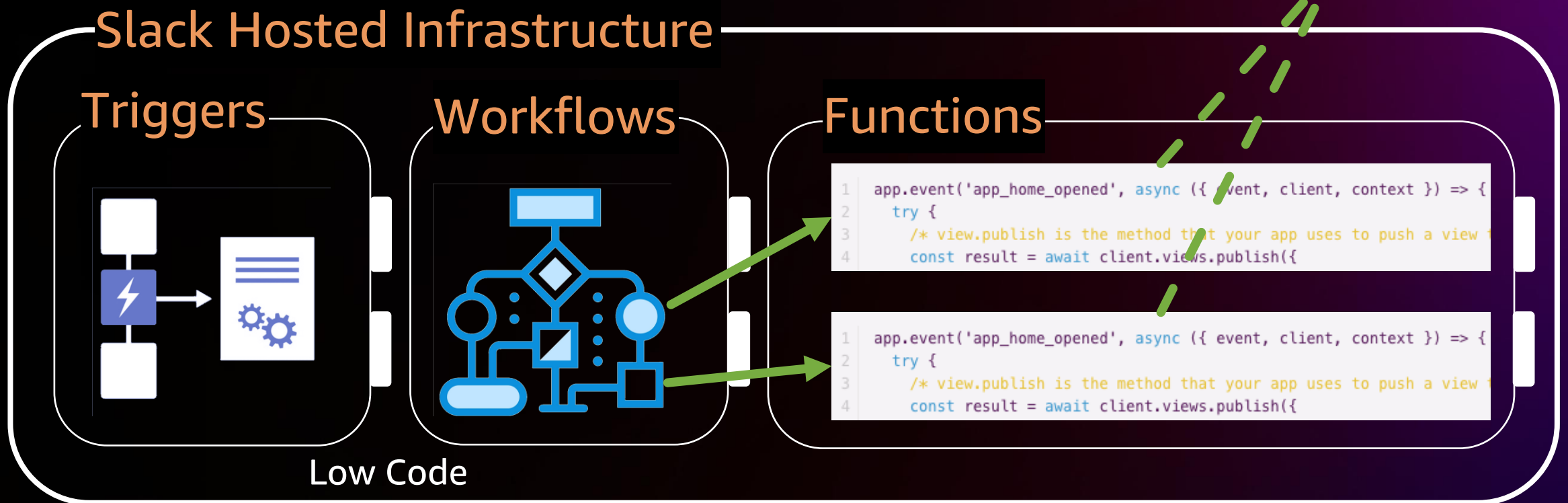


Developers CLI Permissions

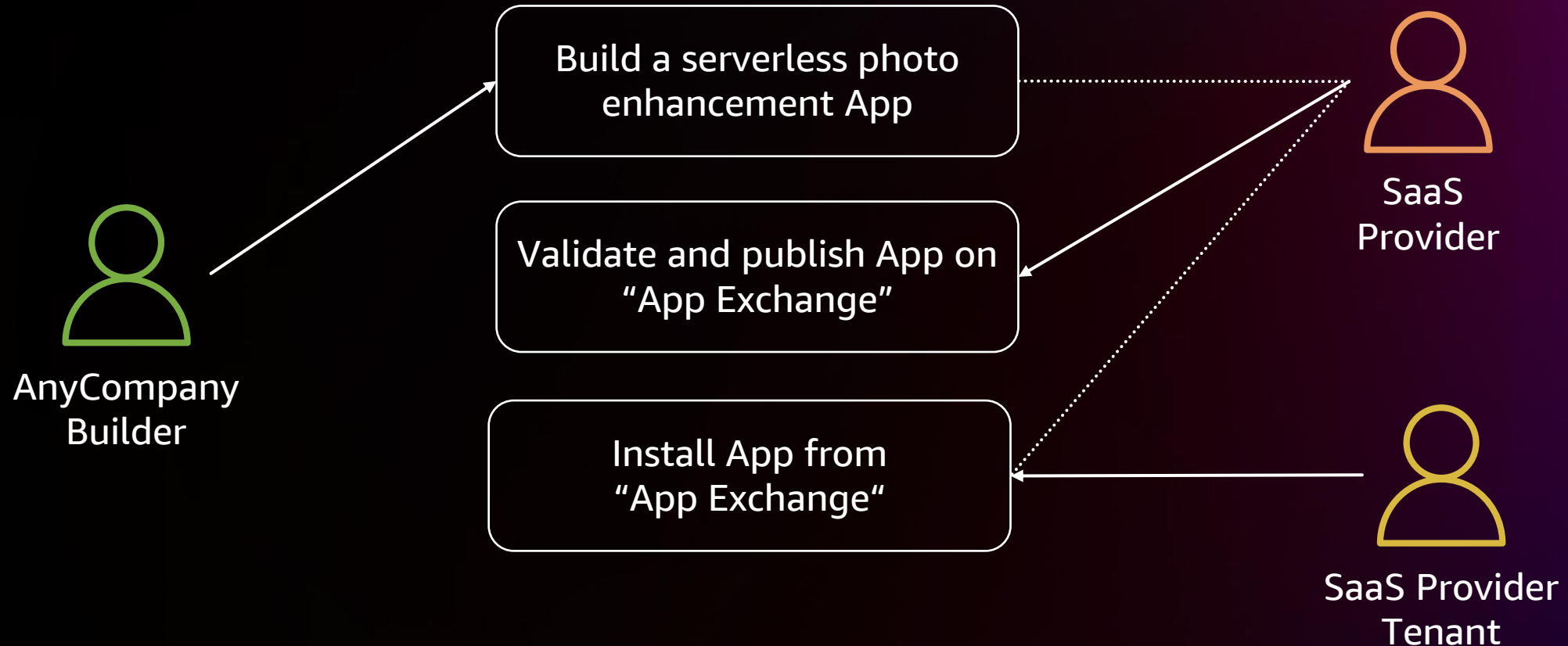
```
(Bill Test) $ stripe apps grant permission "customer_read" "Grant Access..."
```

Slack workflow orchestration

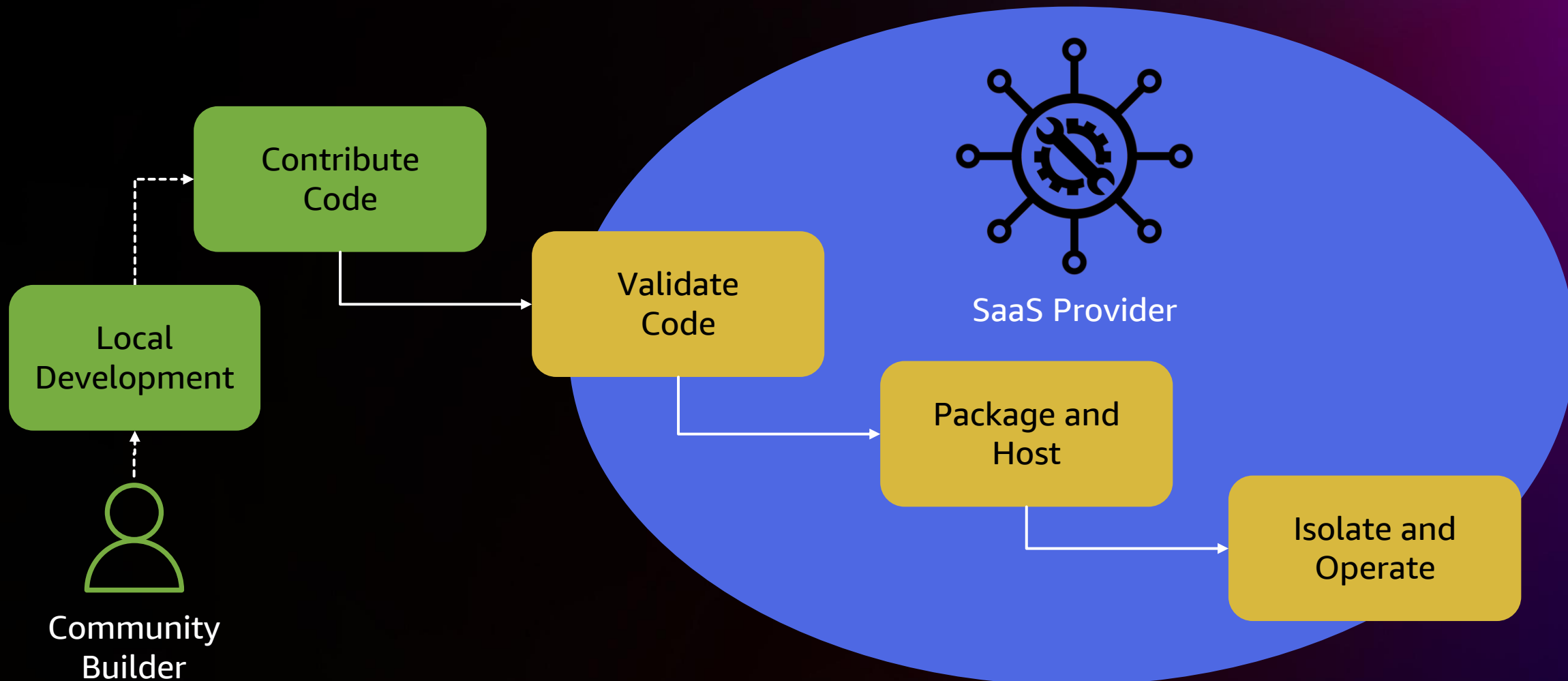
Building Blocks



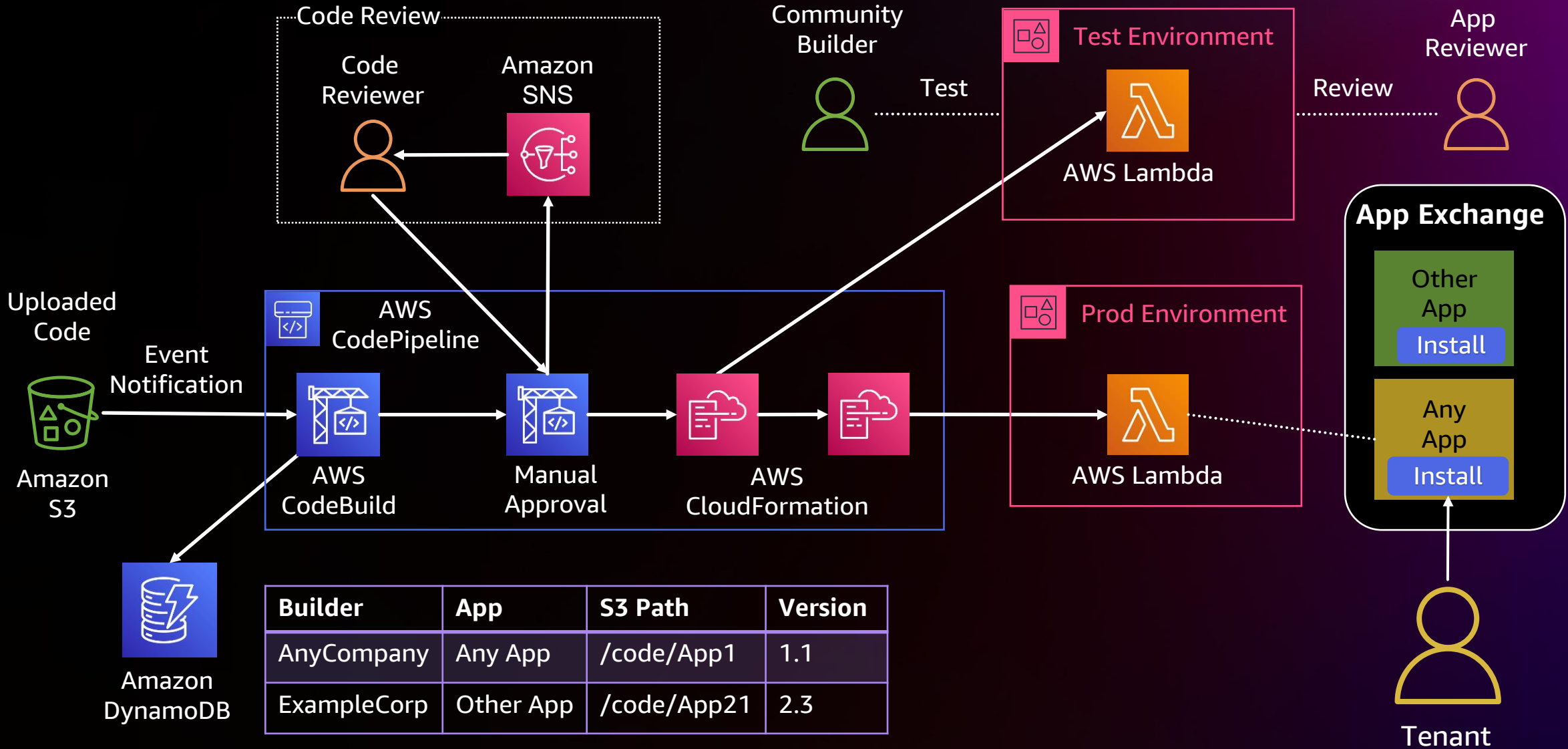
A community extensibility story



Community code contribution

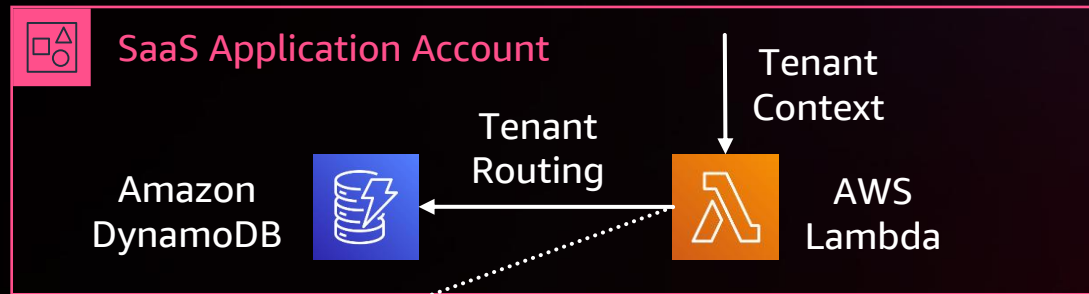


Publishing our app

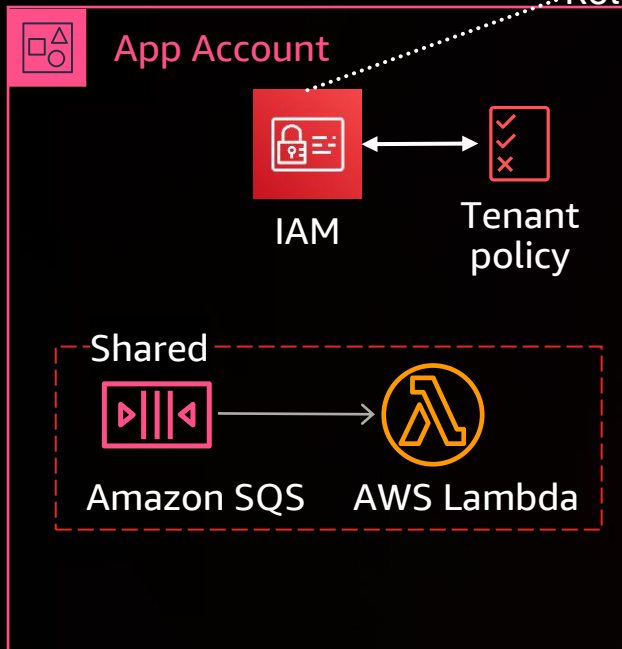


Builder	App	S3 Path	Version
AnyCompany	Any App	/code/App1	1.1
ExampleCorp	Other App	/code/App21	2.3

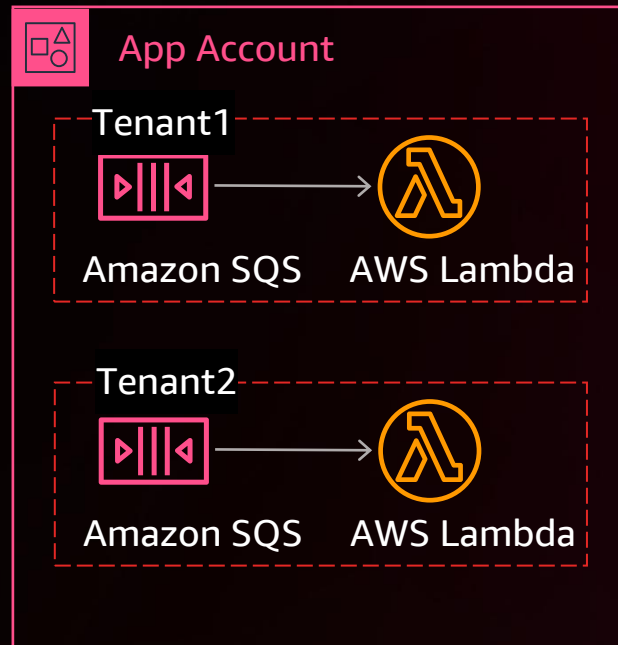
Isolation of serverless app



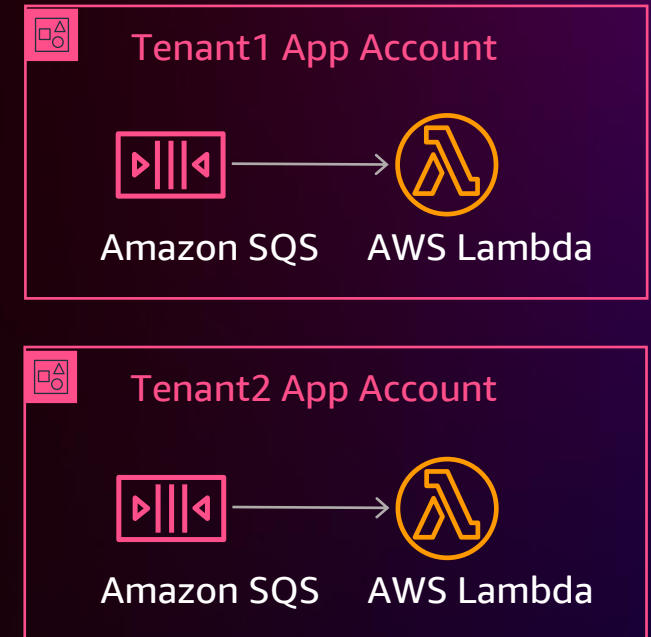
Runtime Isolation



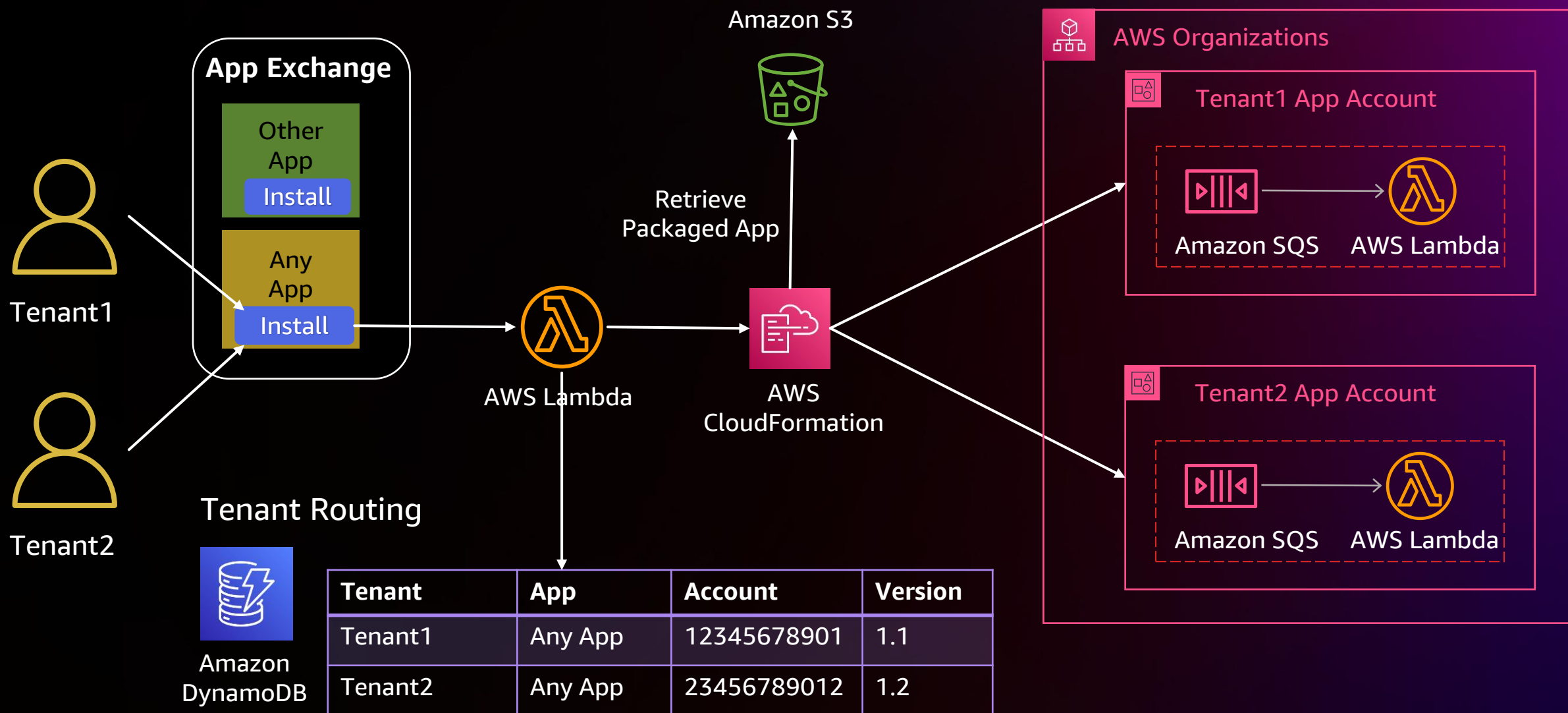
Per Tenant Infrastructure



Account per Tenant



Onboarding tenant per account



Hosting decisions for backend services

SaaS Provider Hosted

- Serverless Functions
- Tenant Isolation
- Observability

App Builder Hosted

- Webhooks
- API Security
- Operational Management

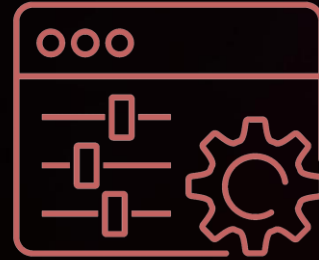
Internal Extensibility



Internal extensibility is different (and not)



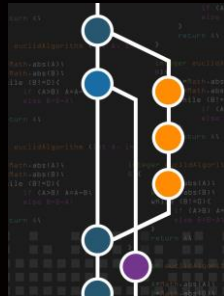
Well-Known
Developers



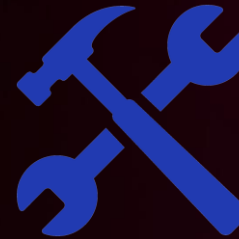
Entitlements/
Features



Simplified
Sandboxing

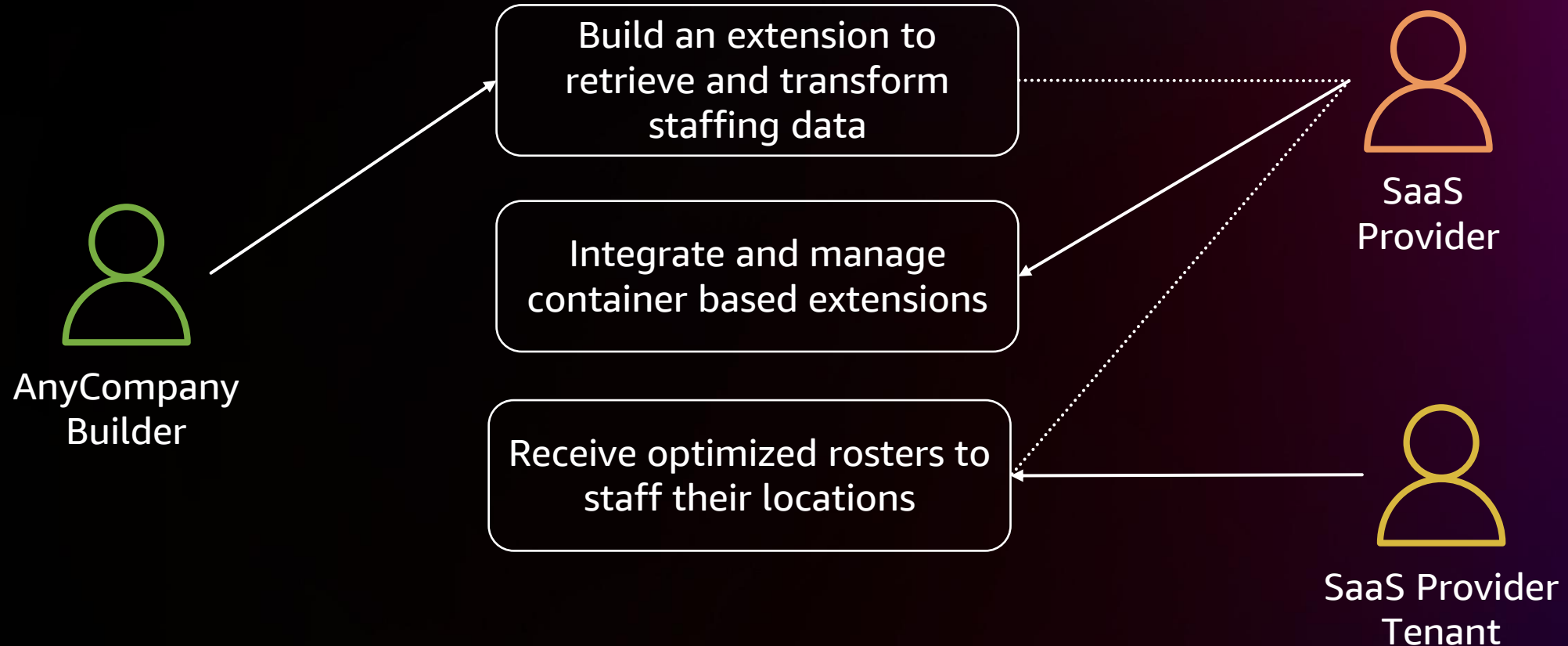


Shared
Lifecycle

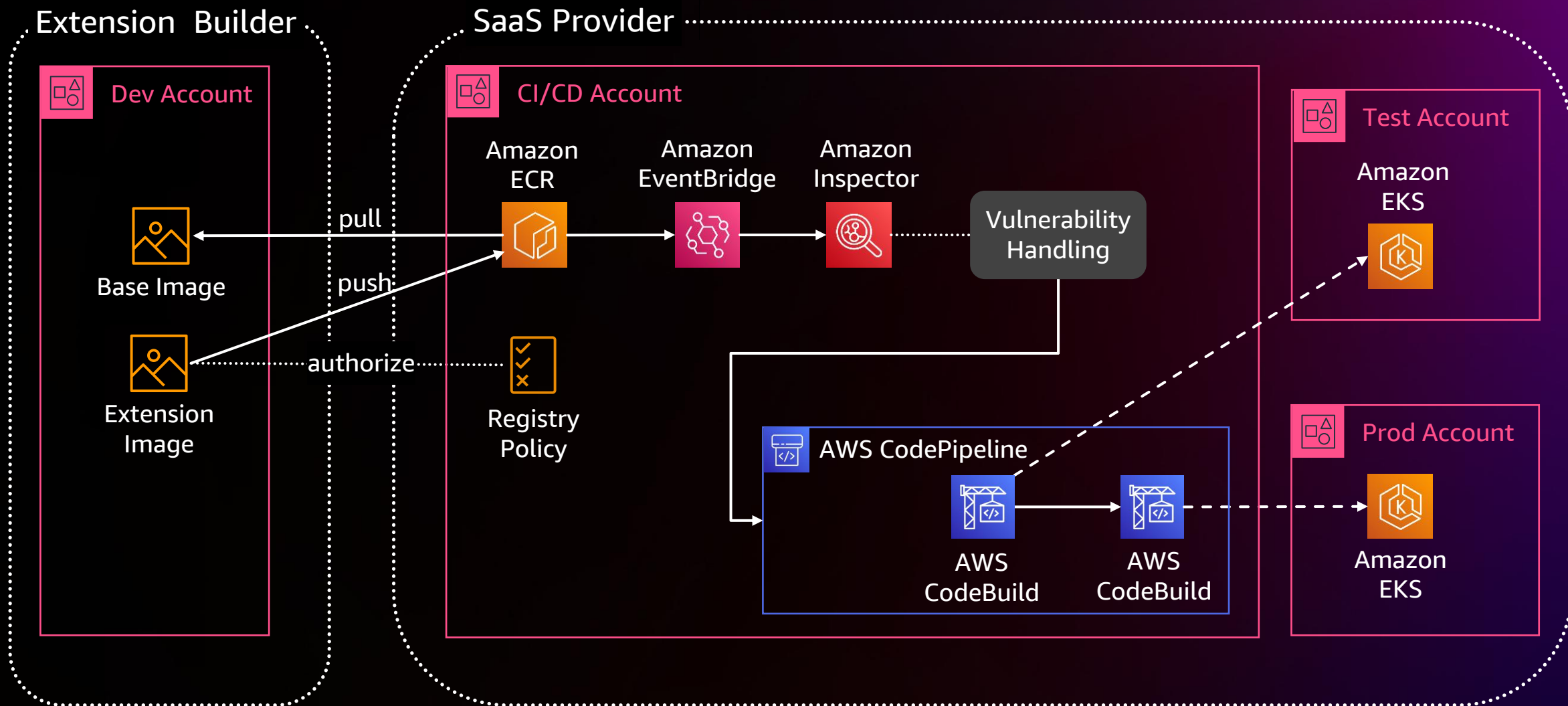


Similar
Toolset

An internal extensibility story

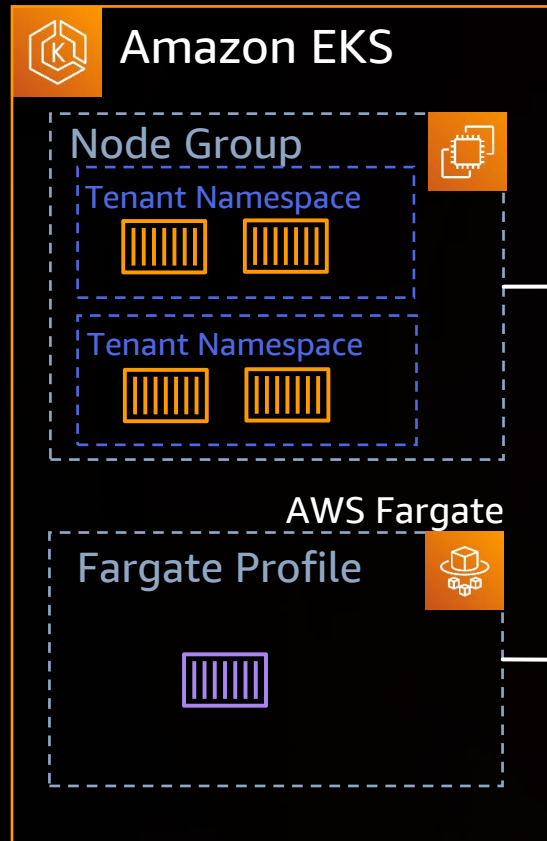


Container code contribution

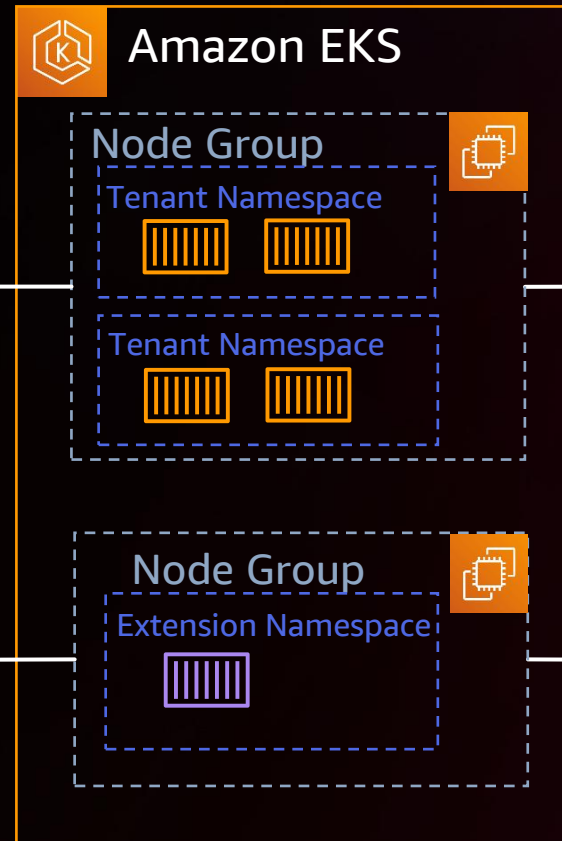


Isolation decisions - tenants and compute

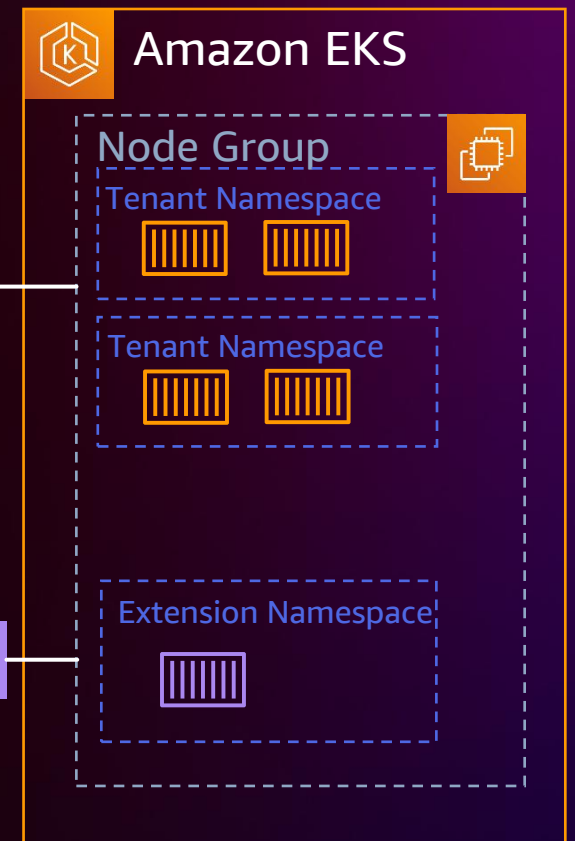
Fargate isolation



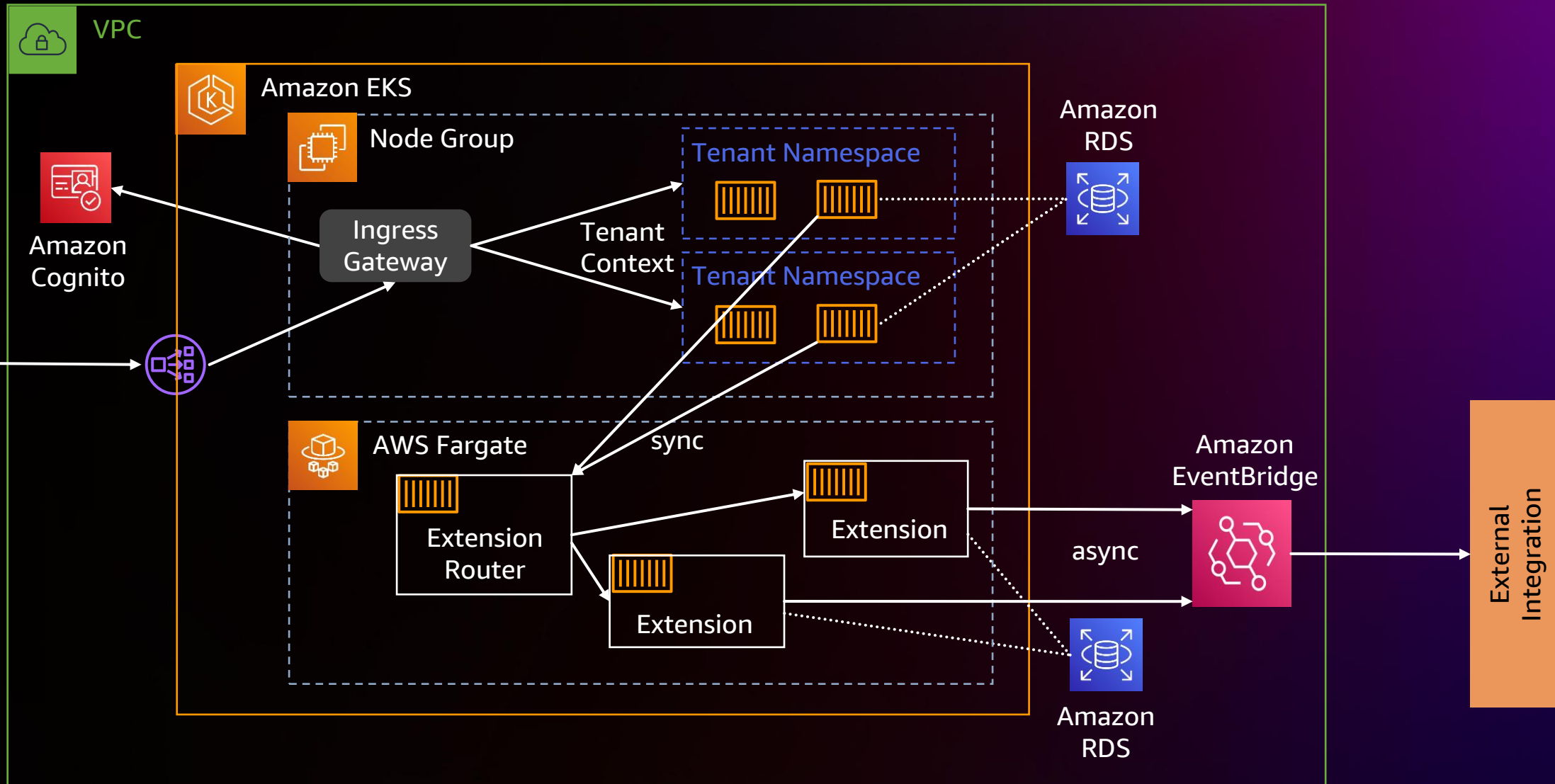
Node Group isolation



Namespace isolation



Container extension example



Extensions Management



Extension management features



Roles and Auditing



Client and Server Access



Extension Metadata



Support Environments

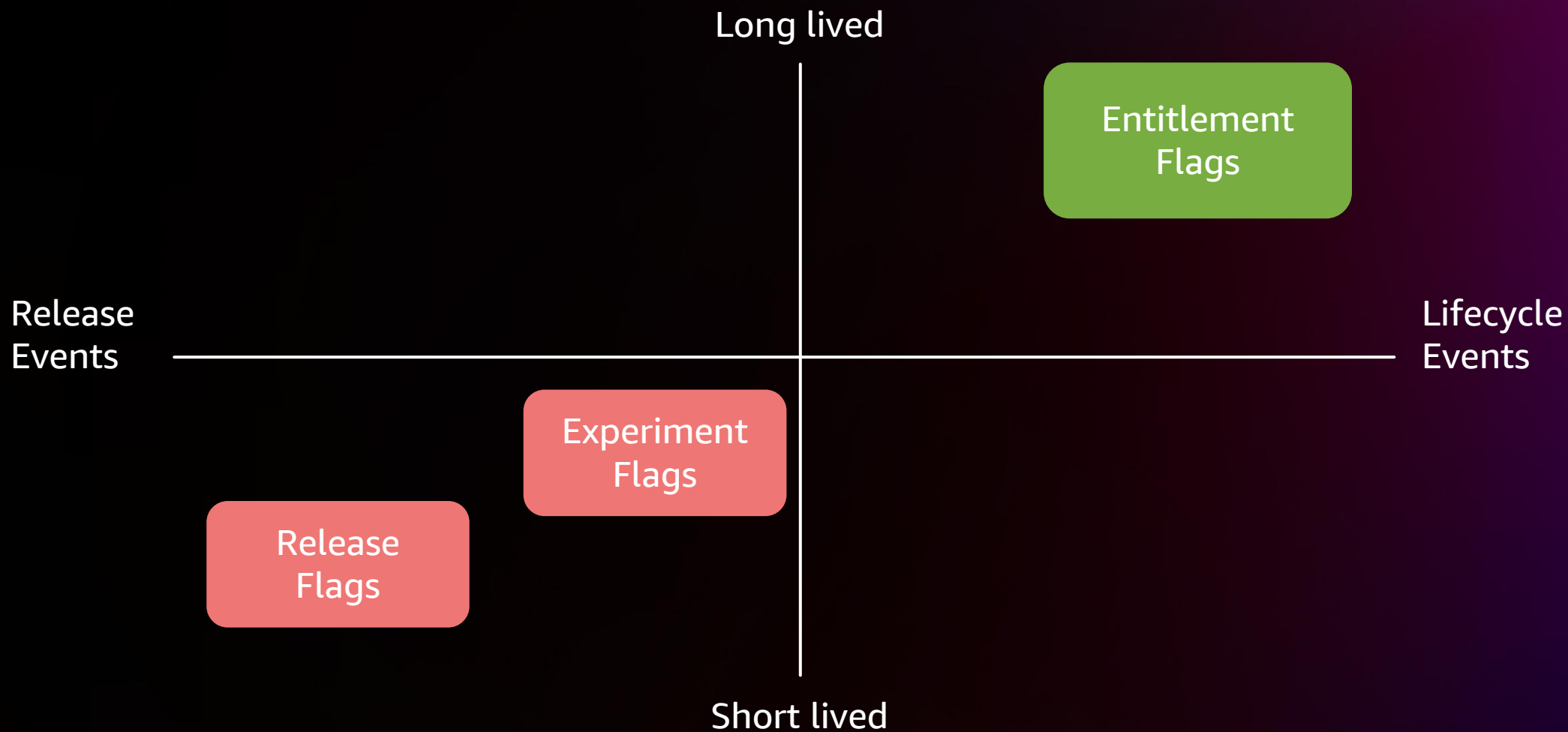


APIs (onboarding)



Performance (Caching)

Managing extensions with flags



Entitlements flags – tenant vs tier

Tenant Feature Targeting



ON Feature 1
TAGS: Tenant1, Tenant28

ON Feature 2
TAGS: Tenant2, Tenant17

Tier Feature Grouping



Standard Tier

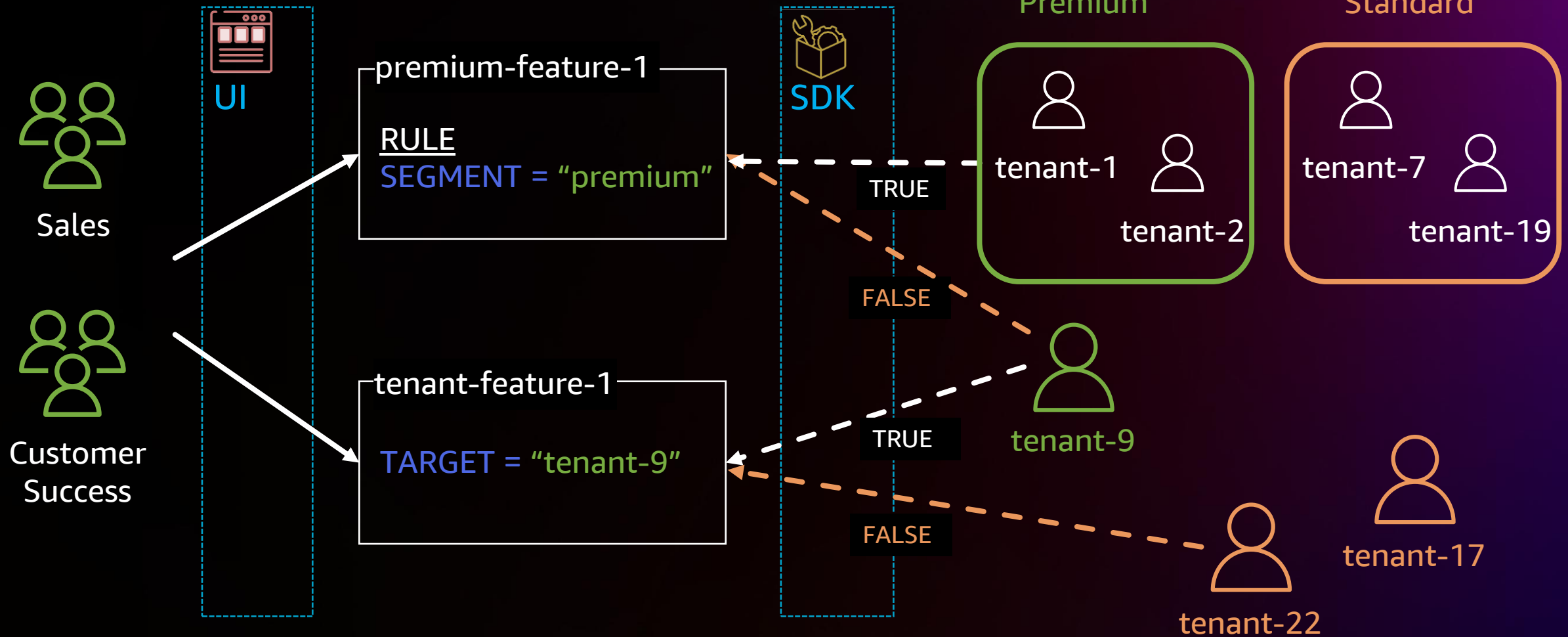
ON Feature 1
OFF Feature 2
OFF Feature 3

Advanced Tier

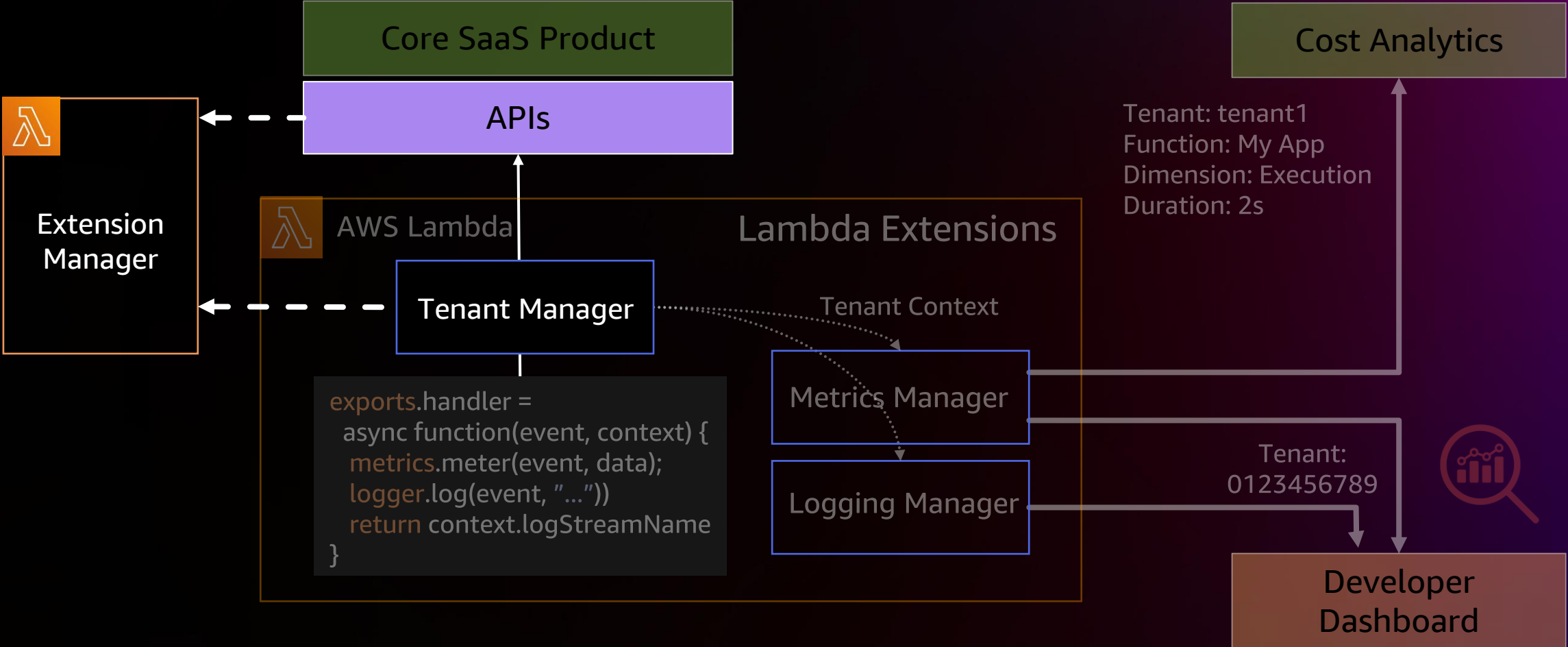
ON Feature 1
ON Feature 2
OFF Feature 3

Entitlements flag example

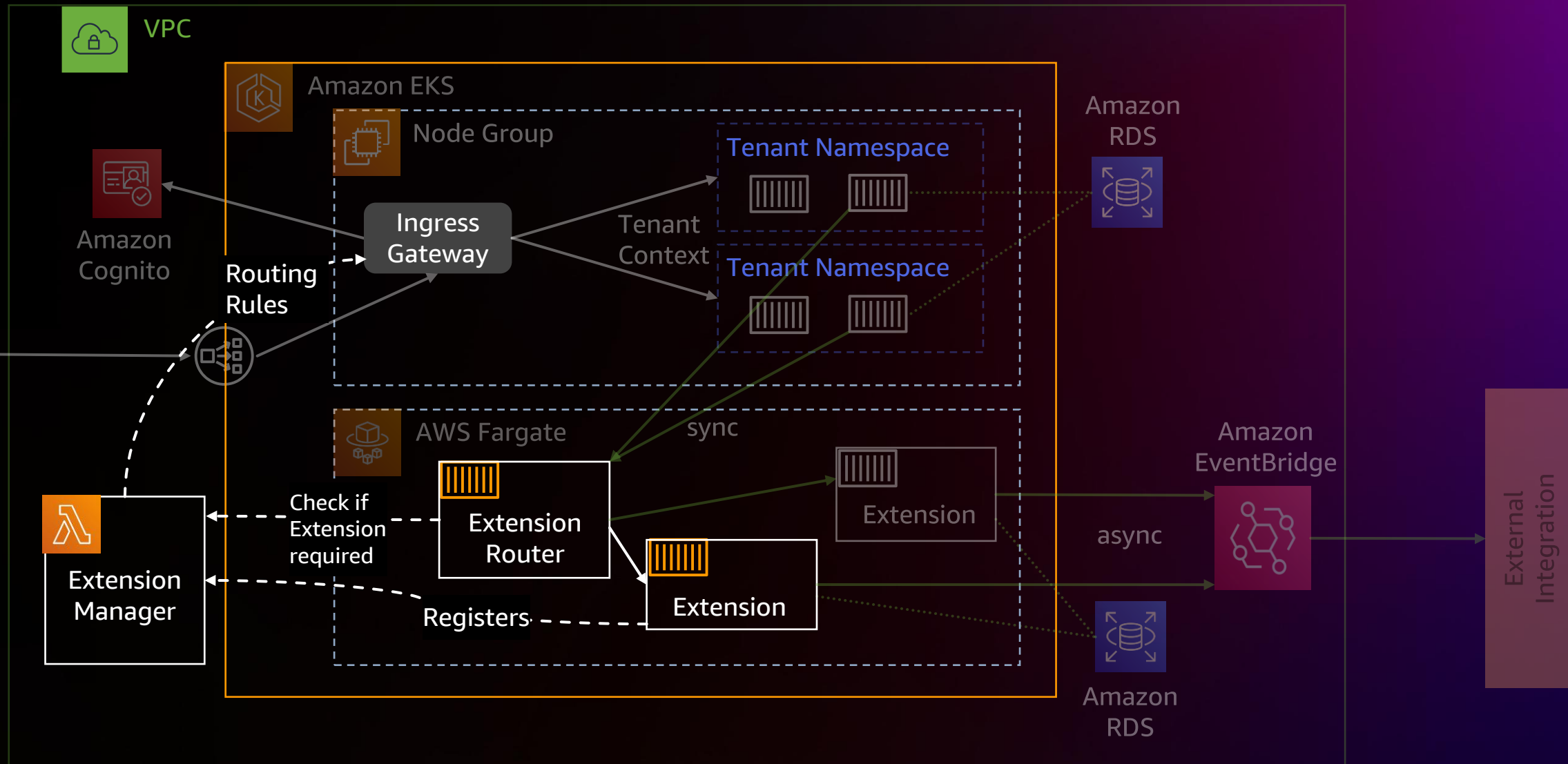
LaunchDarkly →



Serverless app extension manager



Container extension example



Takeaways

- Extensibility is an mechanism to drive growth
- Focus on developer experience
- Tenant isolation and cost take on new dimensions
- Plan for extension management

More SaaS sessions

Breakout sessions

- SAS305 – SaaS architecture patterns: From concept to implementation
- SAS405 – SaaS microservices deep dive: Simplifying multi-tenant development
- SAS306 – SaaS migration: Inside a real-world multi-tenant transformation
- PEX310 – Optimizing your multi-tenant SaaS Architecture

Workshops

- SAS403 - SaaS microservices deep dive: Multi-tenancy meets microservices
- SAS402 – Serverless meets SaaS: Inside a real-world serverless SaaS solution
- SAS401 – Amazon EKS SaaS: Building a working multi-tenant environment

Business session

- PEX209 – Building your SaaS journey on AWS

More SaaS sessions

Chalk talks

- SAS307 – DevOps and SaaS: Applying automation in multi-tenant environments
- SAS303 – SaaS anywhere: Building SaaS solutions that run in hybrid models
- SAS301 – Multi-tenant meets ML: Building ML-based SaaS environments
- SAS304 – Solving the SaaS compliance puzzle
- PEX313 – The SaaS control plane: The heart of SaaS growth
- ARC403 – Amazon EKS SaaS deep dive: Inside a multi-tenant EKS solution
- ARC323 – Designing a multi-tenant SaaS tiering and throttling strategy
- SVS315 - Building multi-tenant applications with AWS Lambda and AWS Fargate

Builder session

- ARC327 – How to optimize cost in your multi-tenant architecture

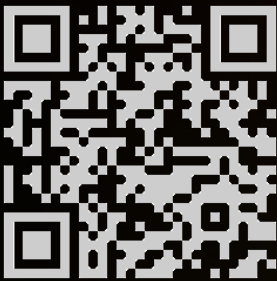


Additional resources

1

Subscribe to AWS SaaS Insights

Get monthly emails with bite-size advice and the latest updates.



2

Explore the SaaS on AWS hub

Check out the SaaS on AWS page for more resources and insights.



3

Discover resources for builders

Access our curated list of SaaS reference solutions, demos, tech events and more.



Thank you!

Bill Tarr

billtarr@amazon.com

@SaaS Tarr



Please complete the session survey in the **mobile app**

