AWS re:Invent

NOV. 29 – DEC. 3, 2021 | LAS VEGAS, NV

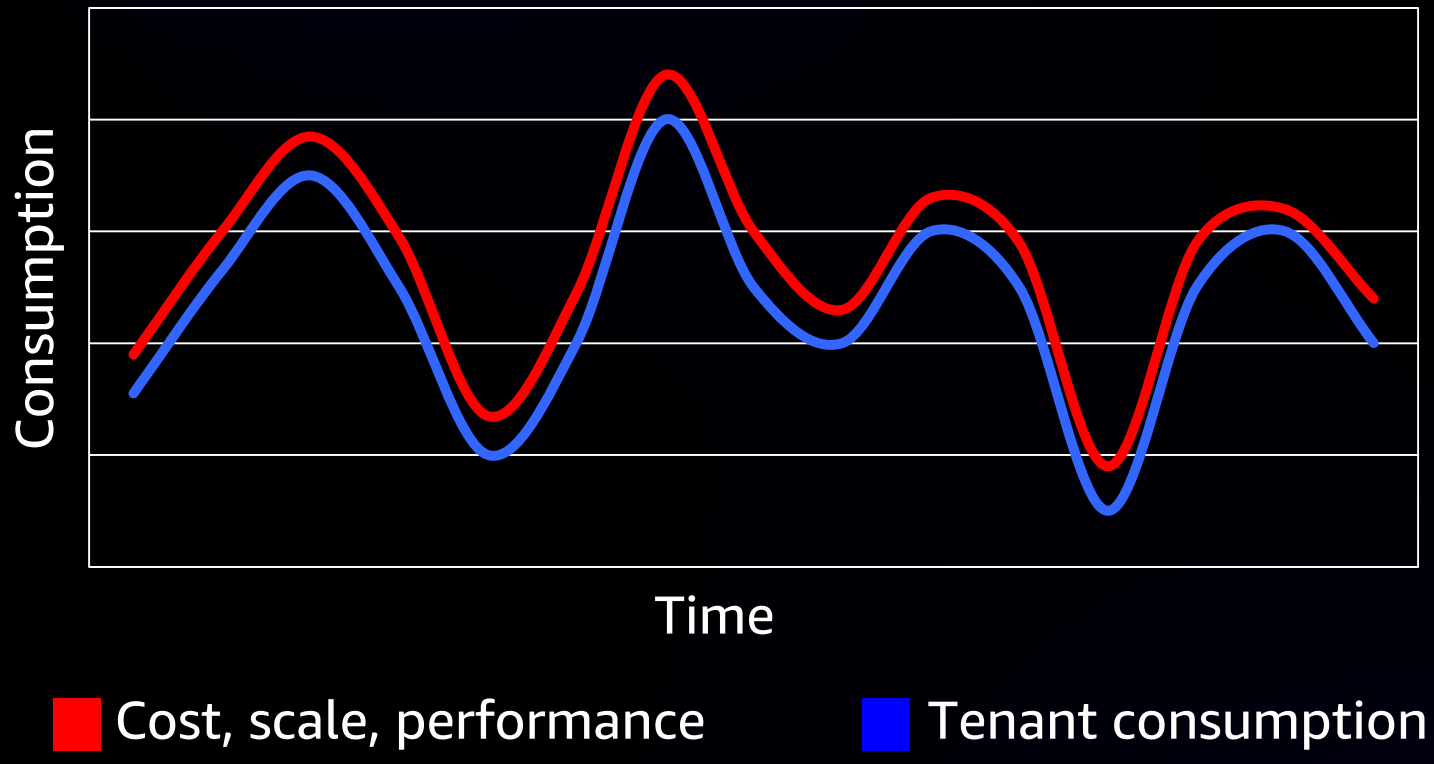
ARC405

Inside a working serverless SaaS reference solution

Tod Golding
Principal Partner Solutions Architect, AWS SaaS Factory
AWS

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The fit between serverless and SaaS

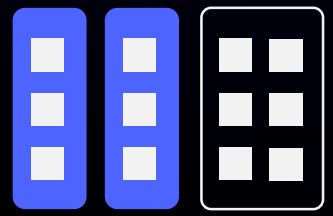


- Agility
- Cost optimization
- Operational efficiency
- Blast radius
- Focus on IP

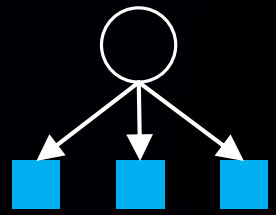
aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

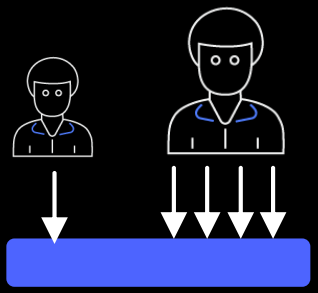
Key architecture considerations



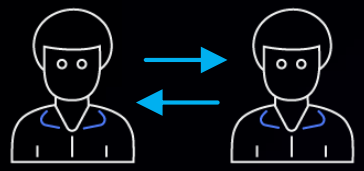
Which deployment models will you support?



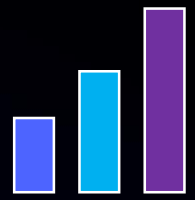
How will you automate tier-based deployments?



How do you manage noisy neighbor conditions?



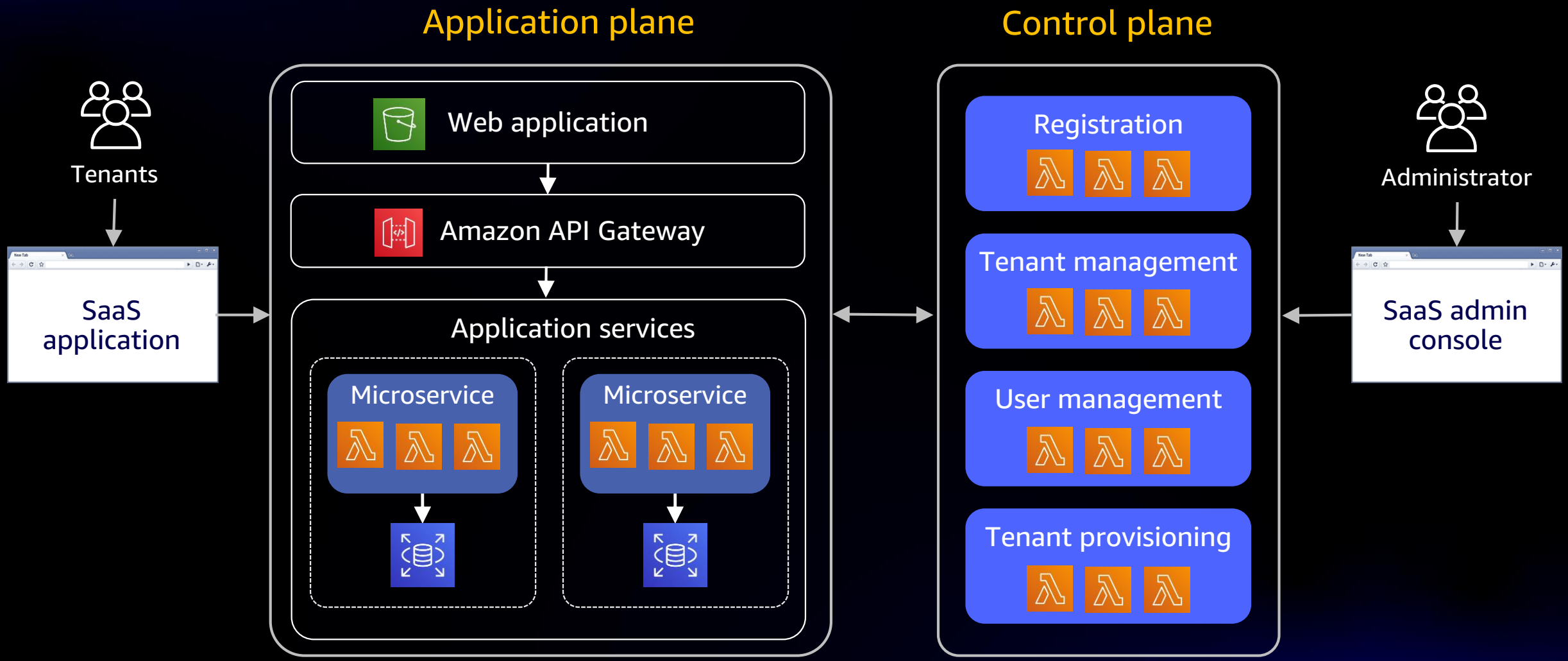
How will you isolate tenant resources?



How will you support tiered tenant experiences?

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

High-level architecture



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Provisioning the control plane

Bootstrap template

Create tables

- Tenant user mapping
- Tenant details
- Tenant stack mapping
- Environment settings

User pools

- Pooled user pool
- Admin user pool

Microservices

- Tenant registration
- Tenant management
- User management

API Gateway

- Routes and permissions
- Tenant registration
- Tenant management
- User management
- Amazon CloudWatch Logs

Web applications

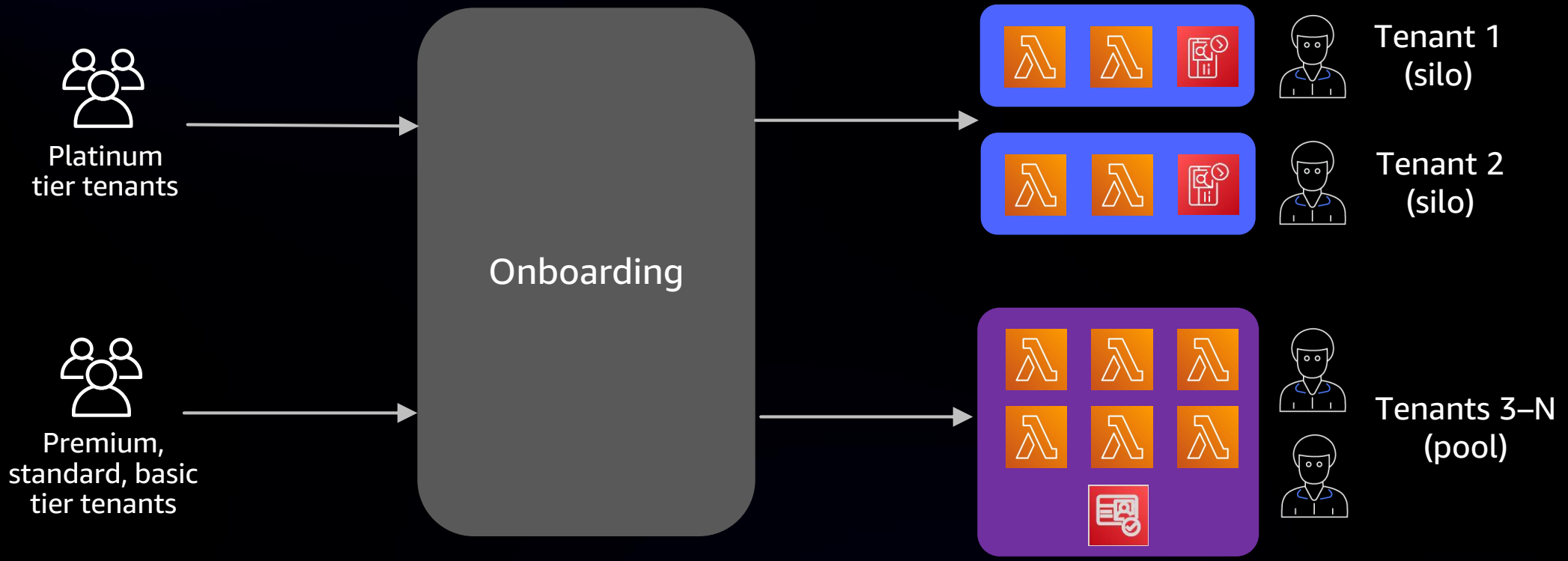
- Admin, SaaS app, landing page sites
- Amazon S3 bucket
- Amazon CloudFront distro

Custom resources

- Update settings tables
- Update tenant stack map

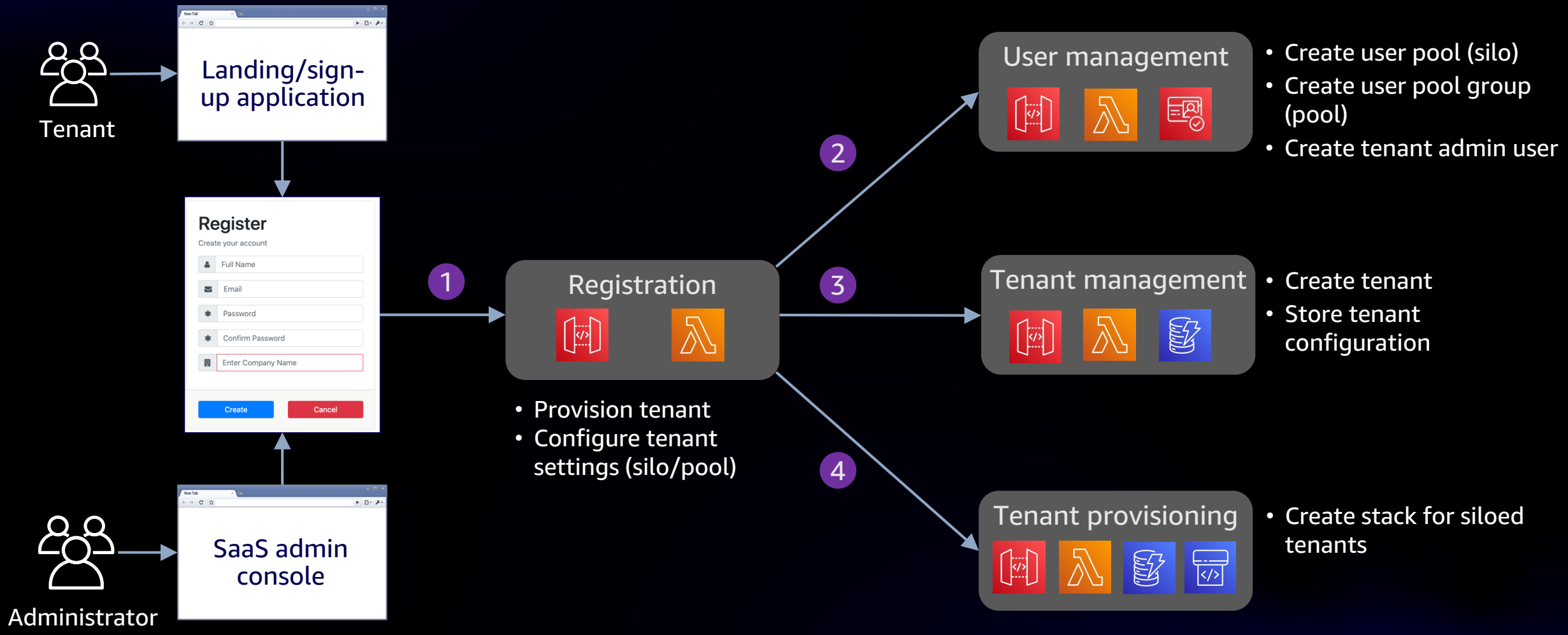
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Two distinct tenant deployment models



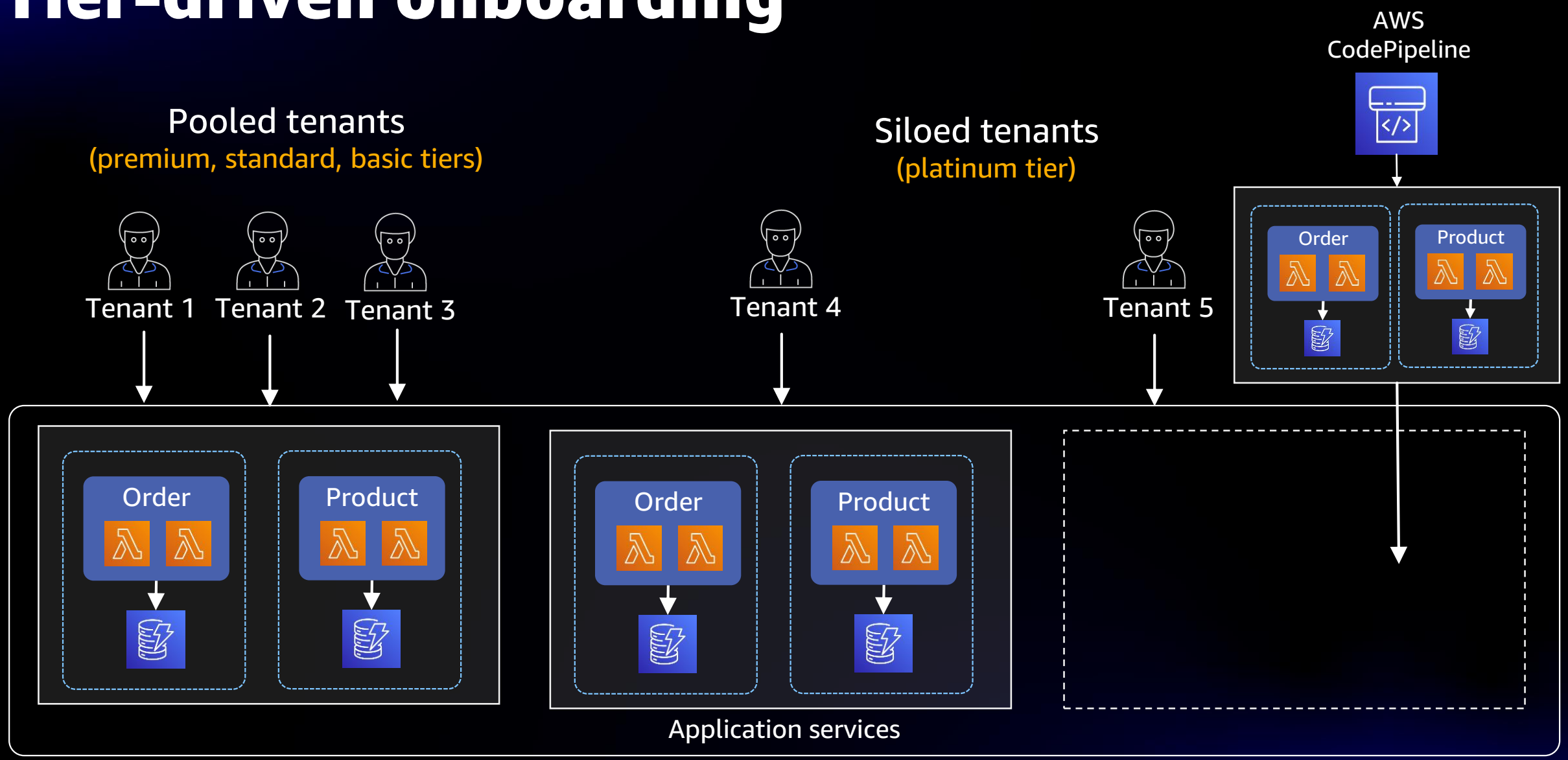
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Registering new tenants



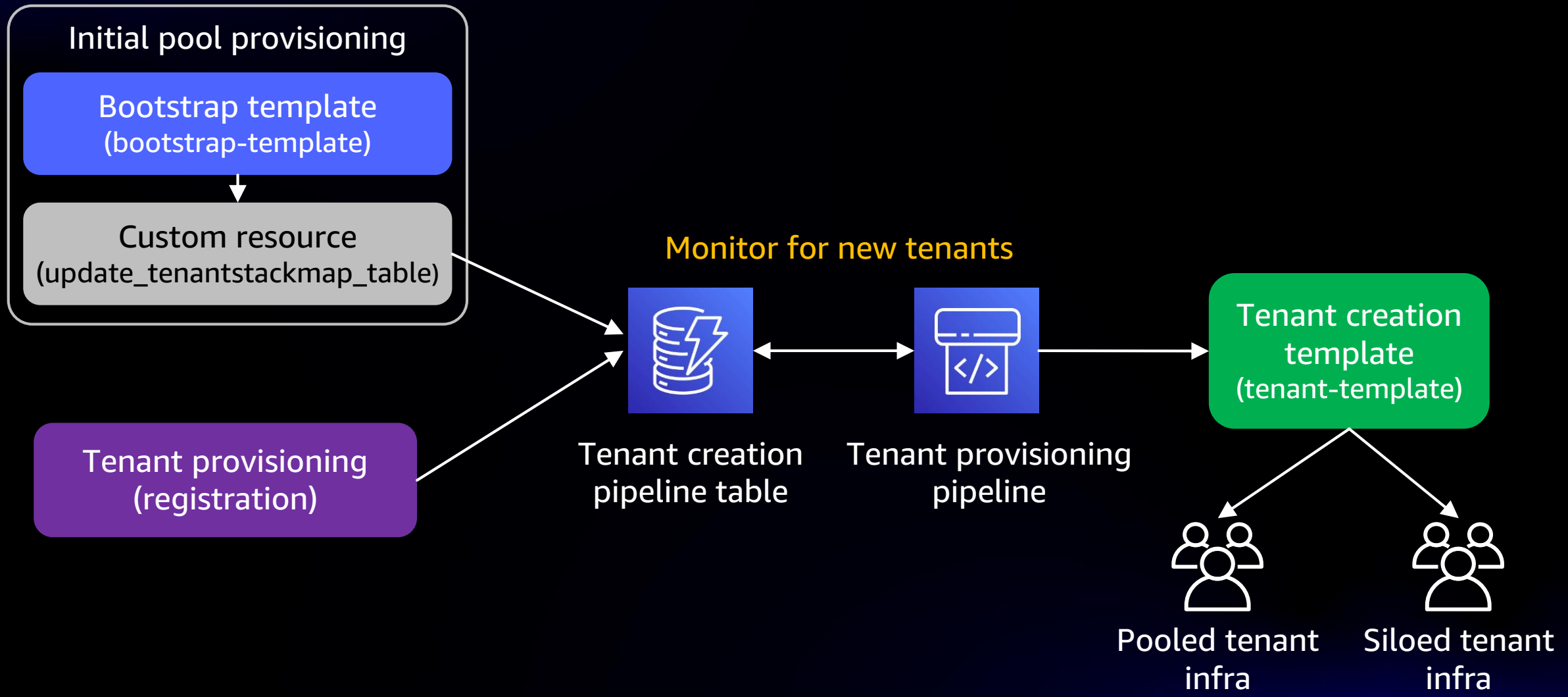
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Tier-driven onboarding



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

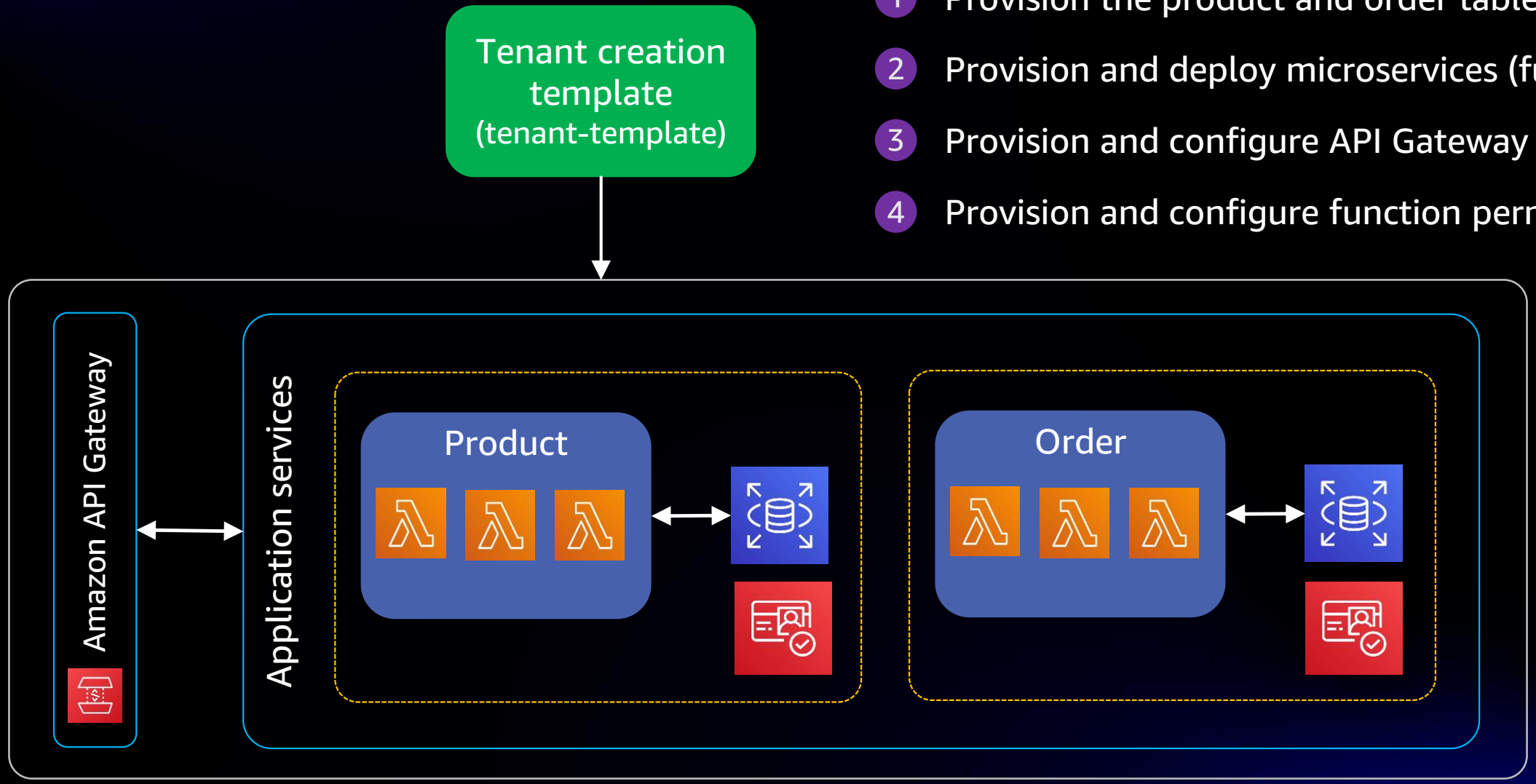
Inside tenant provisioning



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The tenant creation stack

1. Provision the product and order tables
2. Provision and deploy microservices (functions)
3. Provision and configure API Gateway
4. Provision and configure function permissions

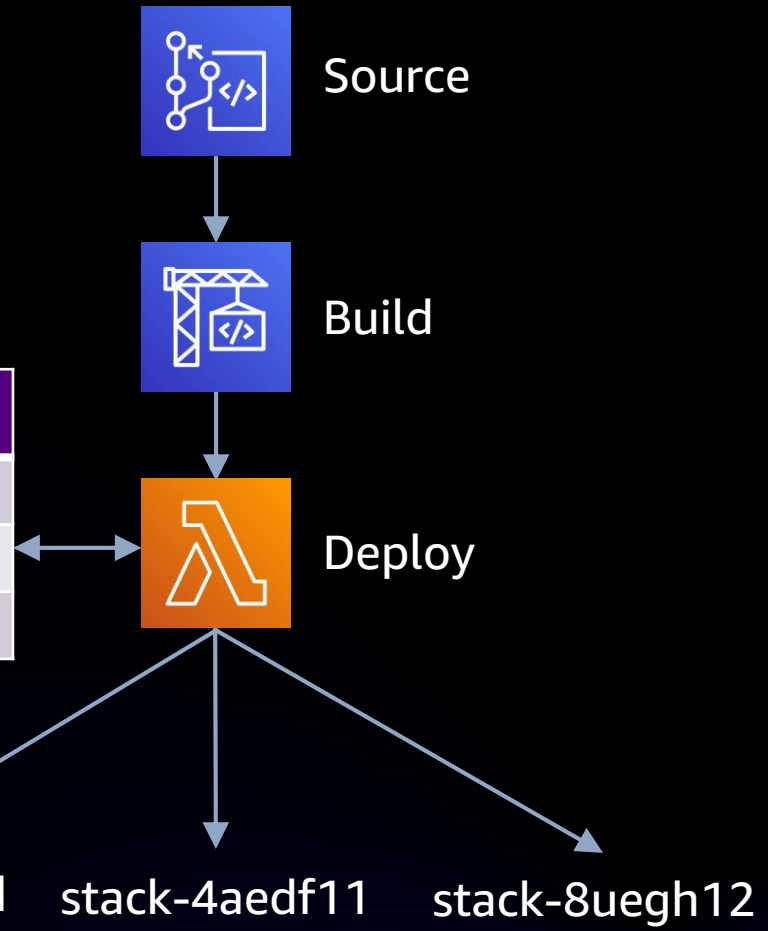


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Mapping the deployments

TenantStackMapping

tenantId	applyLatest	codeCommitId	stackName
4aedf11	true	0a65d	stack-4aedf11
8uegh12	true	0a65d	stack-8uegh12
pooled	true	0a65d	stack-pooled



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Applying tiers and configuring tenants

tenant-registration.py - register_tenant()

```
34      if (tenant_details['tenantTier'].upper() == utils.TenantTier.PLATINUM.value.upper()):
35          tenant_details['dedicatedTenancy'] = 'true'
36          api_key = platinum_tier_api_key
37      elif (tenant_details['tenantTier'].upper() == utils.TenantTier.PREMIUM.value.upper()):
38          api_key = premium_tier_api_key
39      elif (tenant_details['tenantTier'].upper() == utils.TenantTier.STANDARD.value.upper()):
40          api_key = standard_tier_api_key
41      elif (tenant_details['tenantTier'].upper() == utils.TenantTier.BASIC.value.upper()):
42          api_key = basic_tier_api_key
```

```
53      create_user_response = __create_tenant_admin_user(tenant_details, headers, auth, host, stage_name)
54
55      tenant_details['tenantId'] = tenant_id
56      tenant_details['apiKey'] = api_key
57      tenant_details['userPoolId'] = create_user_response['message']['userPoolId']
58      tenant_details['appClientId'] = create_user_response['message']['appClientId']
59      tenant_details['tenantAdminUserName'] = create_user_response['message']['tenantAdminUserName']
60
61      create_tenant_response = __create_tenant(tenant_details, headers, auth, host, stage_name)
```

aws

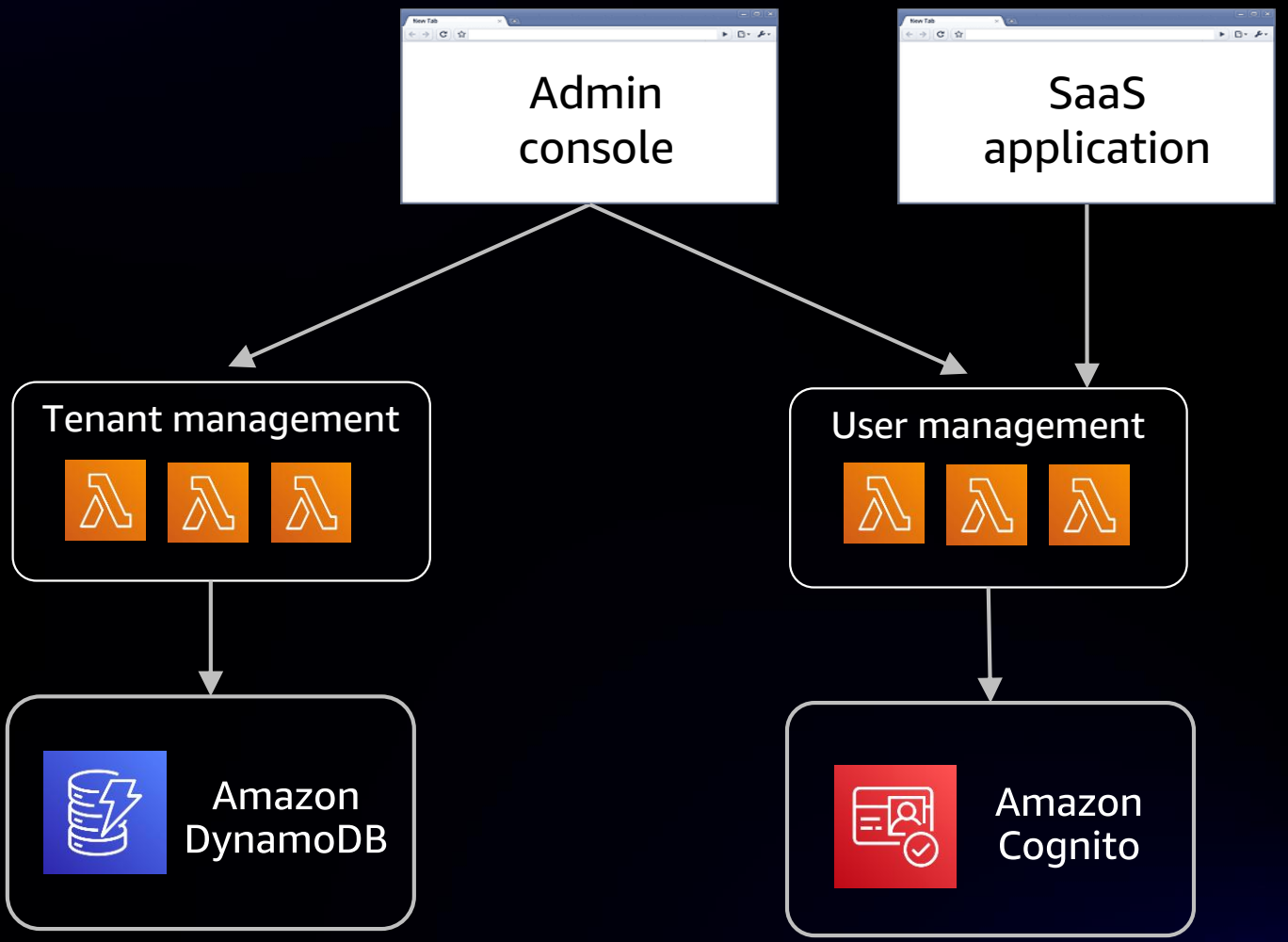
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Triggering the tenant provisioning

```
22  def provision_tenant(event, context):
23      tenant_details = json.loads(event['body'])
24      try:
25          response_ddb = table_tenant_stack_mapping.put_item(
26              Item={
27                      'tenantId': tenant_details['tenantId'],
28                      'stackName': stack_name.format(tenant_details['tenantId']),
29                      'applyLatestRelease': True,
30                      'codeCommitId': ''
31                  }
32              )
33
34          response_codepipeline = codepipeline.start_pipeline_execution(
35              name='serverless-saas-pipeline'
36          )
37      except Exception as e:
38          raise
39      else:
40          return utils.create_success_response("Tenant Provisioning Started")
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Managing tenants and users



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Creating the user and tenant

create_user

```
response = client.admin_create_user(
    Username=user_details['userName'],
    UserPoolId=user_pool_id,
    ForceAliasCreation=True,
    UserAttributes=[
        {
            'Name': 'email',
            'Value': user_details['userEmail']
        },
        {
            'Name': 'custom:userRole',
            'Value': user_details['userRole']
        },
        {
            'Name': 'custom:tenantId',
            'Value': user_tenant_id
        }
    ]
)
```

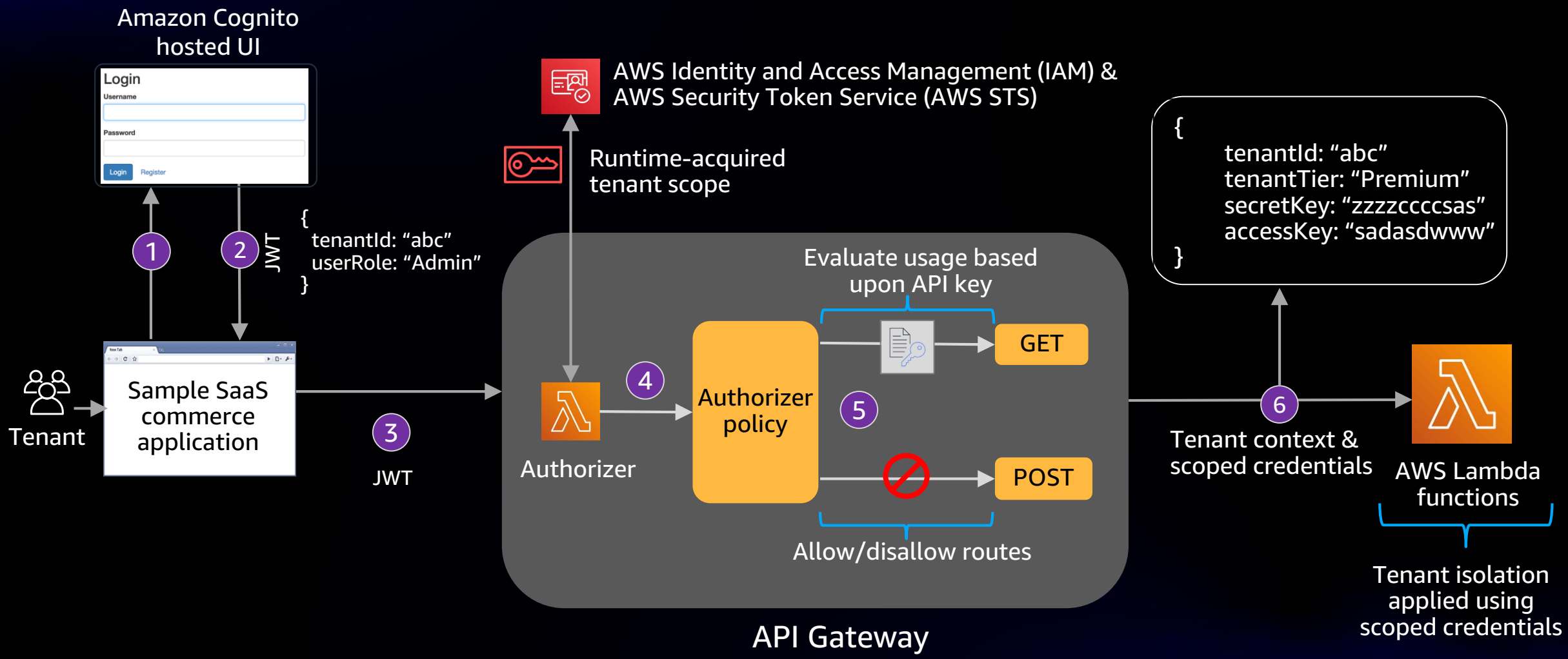
create_tenant

```
if(tenant_details['dedicatedTenancy'].lower()!= 'true'):
    settings_response = table_system_settings.get_item(
        Key={
            'settingName': 'apiGatewayUrl-Pooled'
        }
    )
    api_gateway_url = settings_response['Item']['settingValue']

response = table_tenant_details.put_item(
    Item={
            'tenantId': tenant_details['tenantId'],
            'tenantName' : tenant_details['tenantName'],
            'tenantAddress': tenant_details['tenantAddress'],
            'tenantEmail': tenant_details['tenantEmail'],
            'tenantPhone': tenant_details['tenantPhone'],
            'tenantTier': tenant_details['tenantTier'],
            'apiKey': tenant_details['apiKey'],
            'userPoolId': tenant_details['userPoolId'],
            'appClientId': tenant_details['appClientId'],
            'dedicatedTenancy': tenant_details['dedicatedTenancy'],
            'isActive': True,
            'apiGatewayUrl': api_gateway_url
        }
    )
```

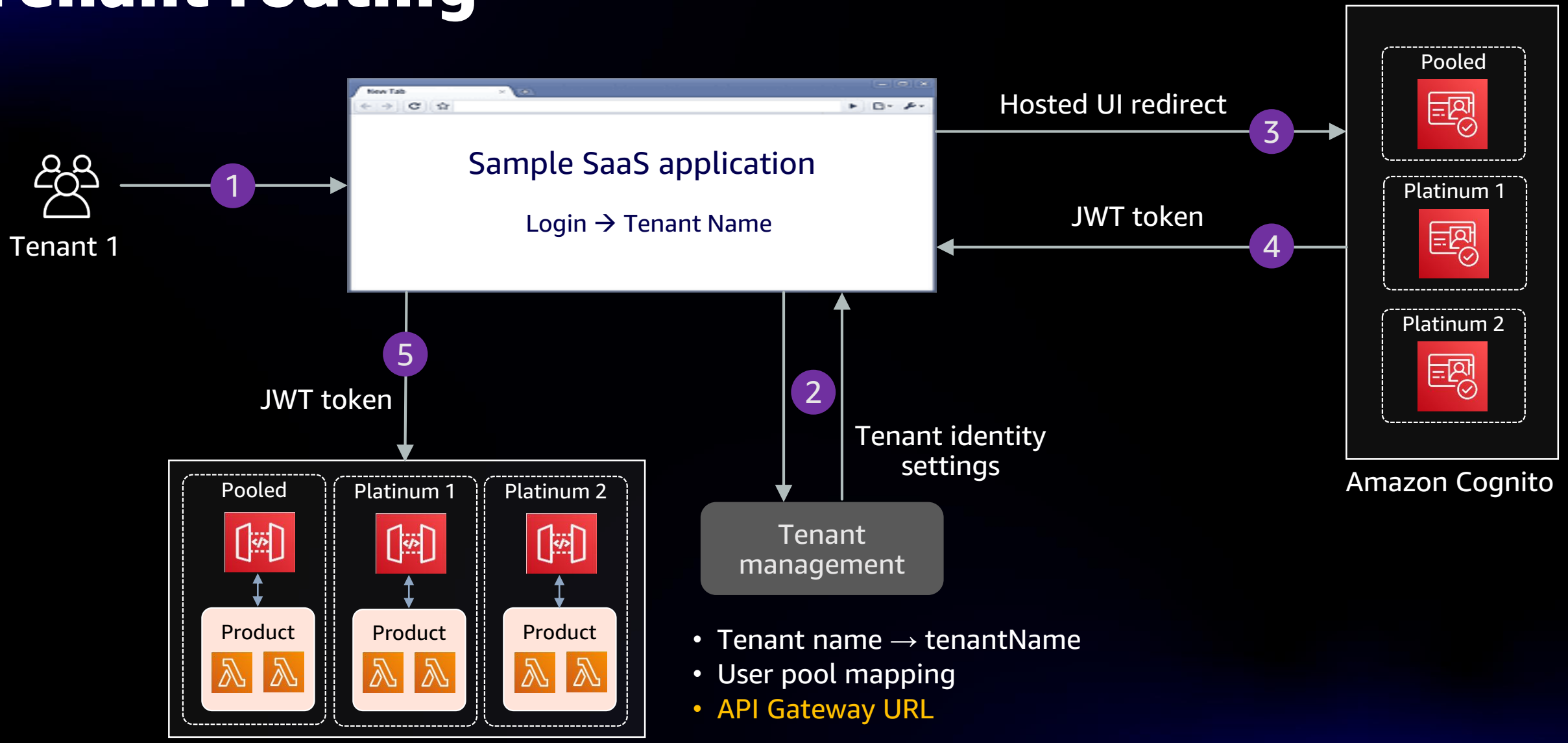
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Authentication and authorization



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Tenant routing

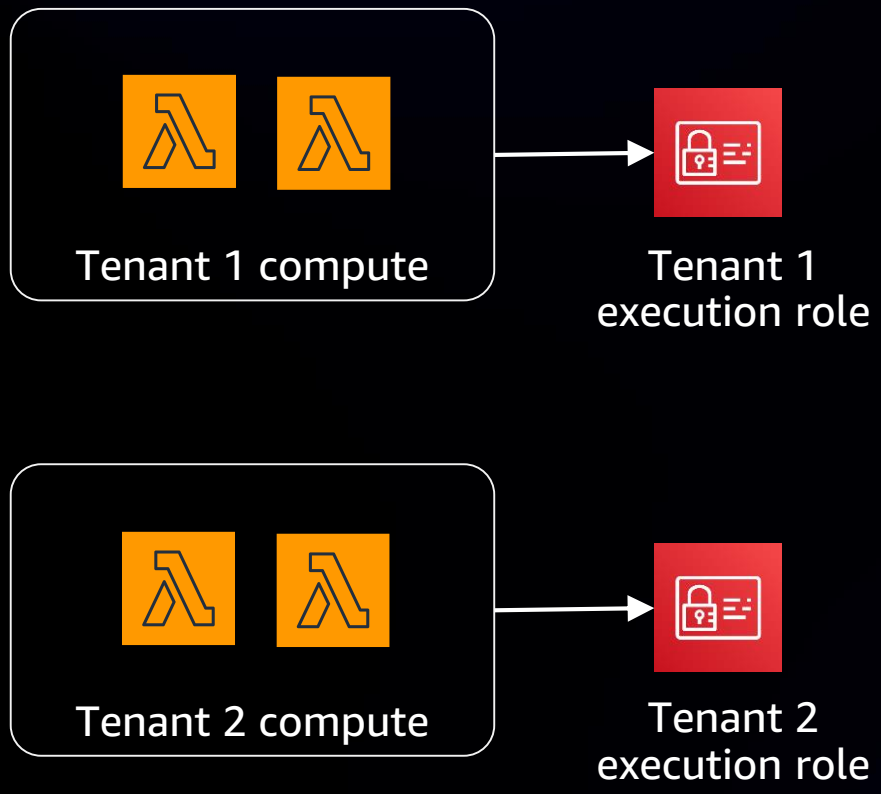


- Tenant name → tenantName
- User pool mapping
- API Gateway URL

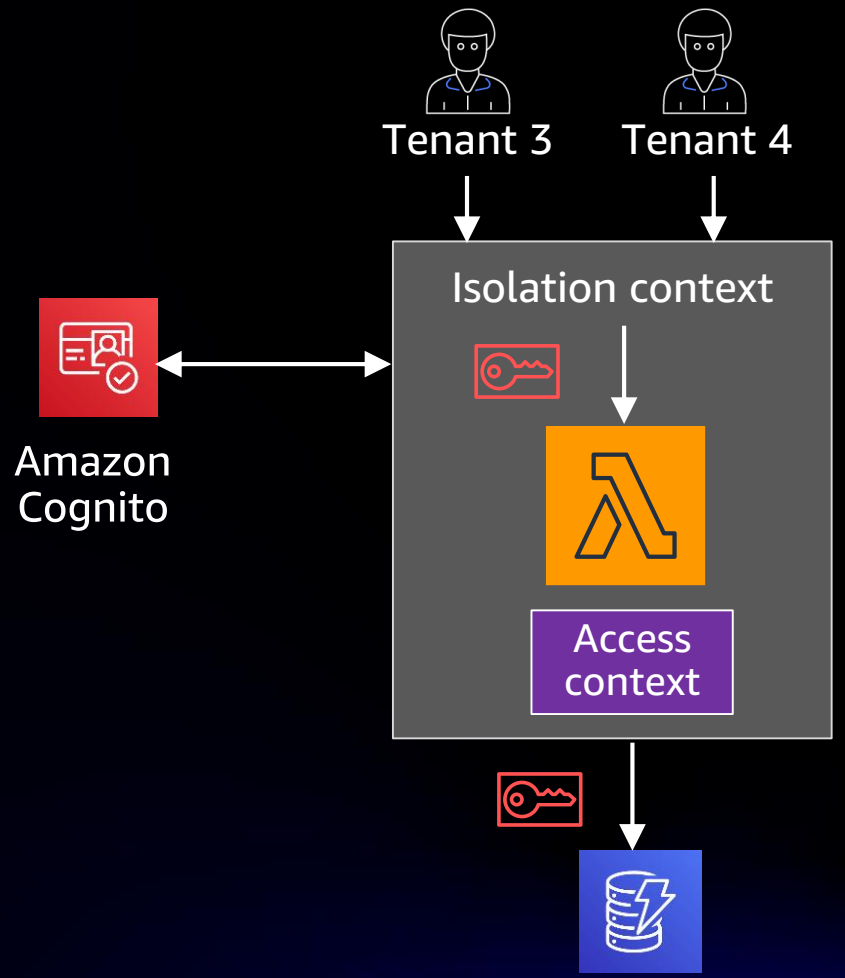
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Two isolation models

Siloed model

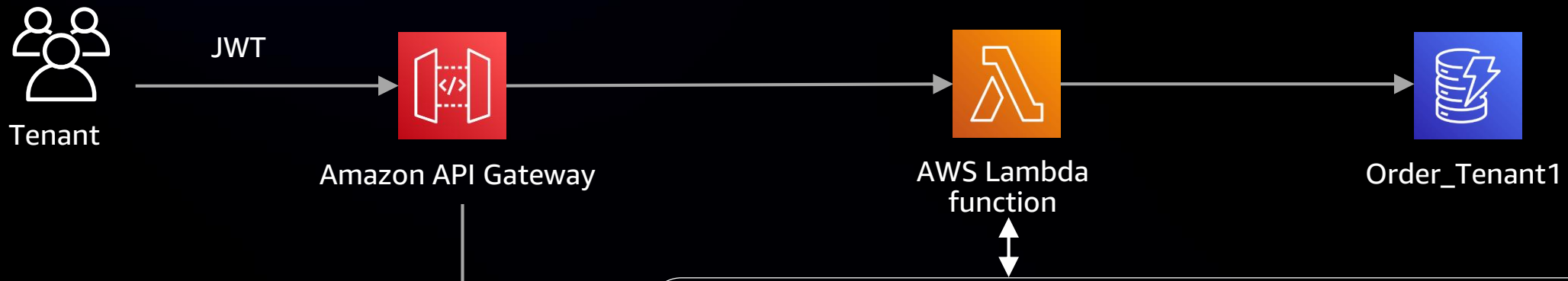


Pooled model



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Silo: Execution role applied during deployment

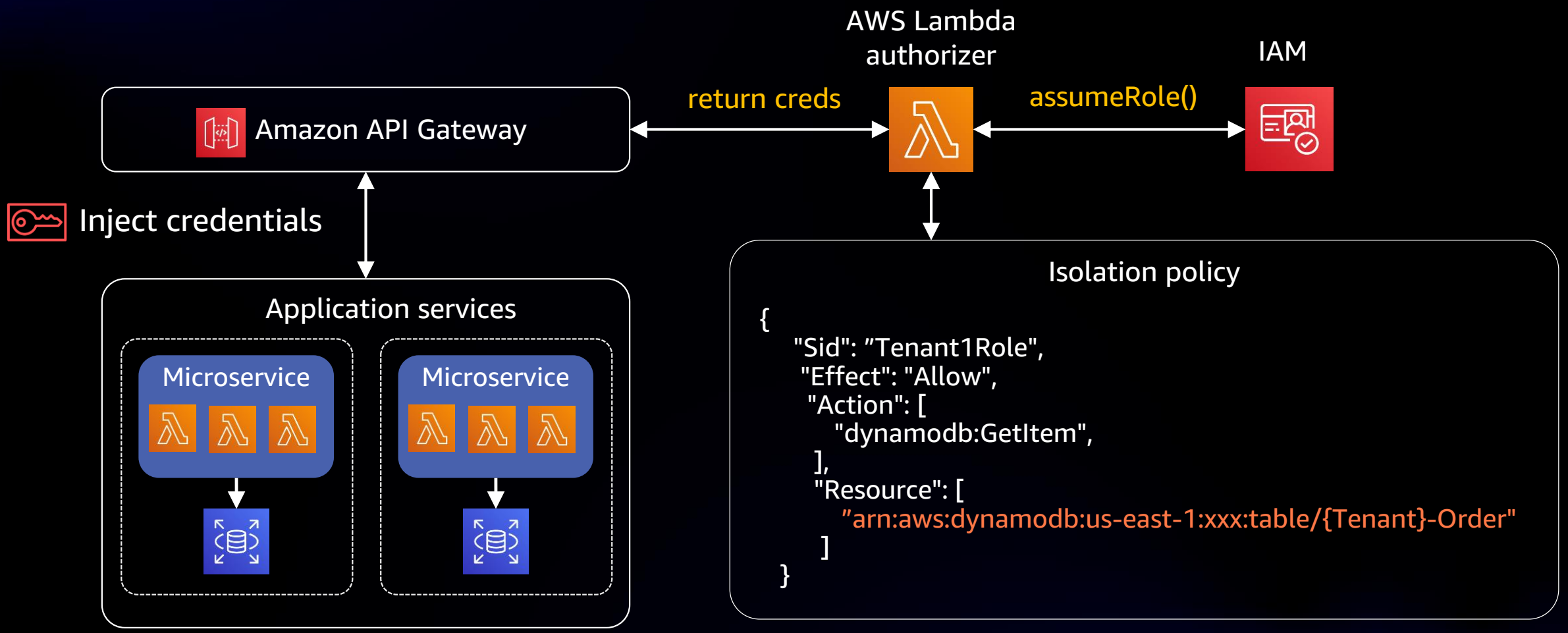


Lambda execution role

```
{
    "Sid": ”Tenant1Role",
     "Effect": "Allow",
      "Action": [
         "dynamodb:GetItem",
      ],
       "Resource": [
          ”arn:aws:dynamodb:us-east-1:xxx:table/Tenant1-Order"
       ]
  }
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Pool: Authorizer-injected credentials



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

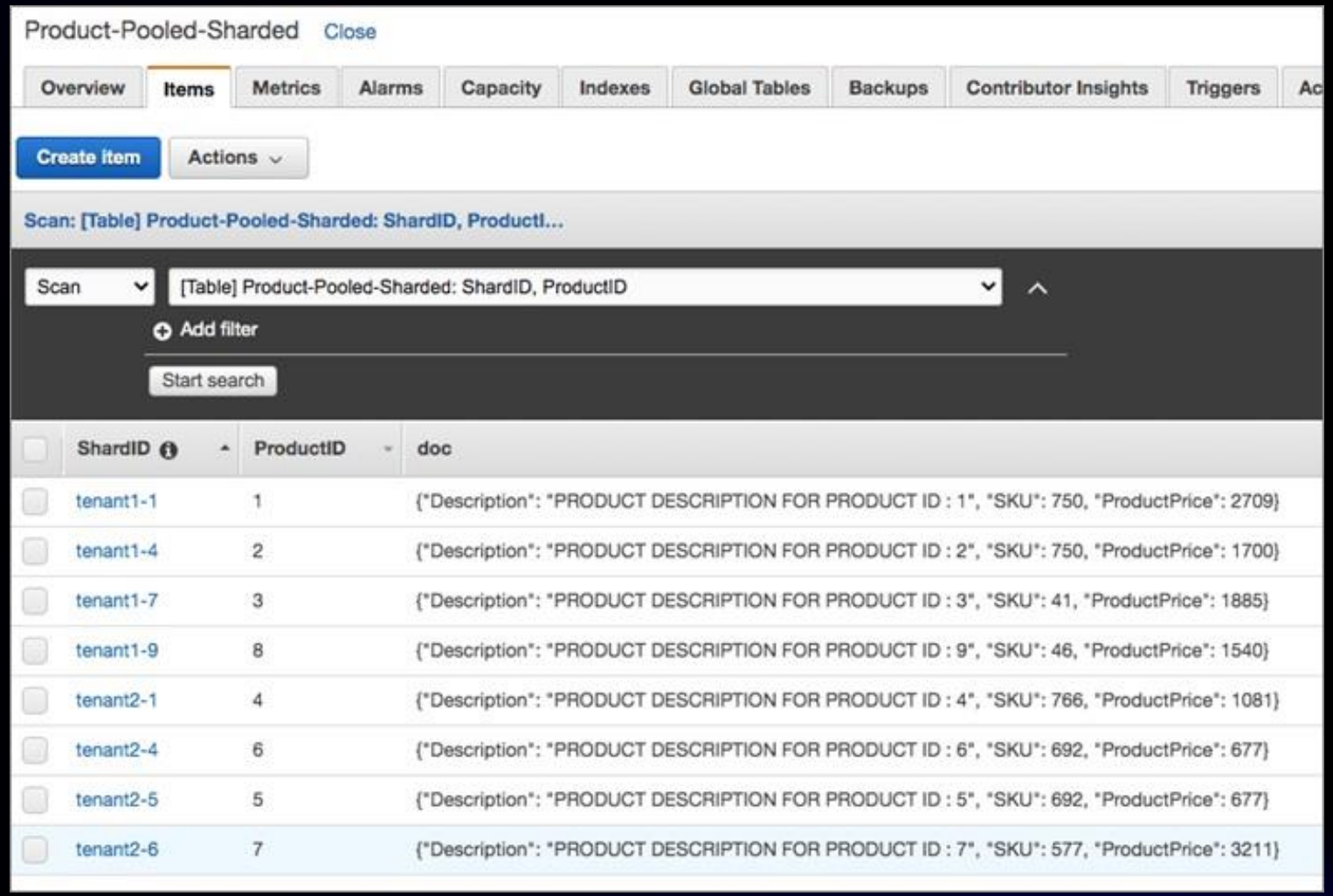
Tier applied in data access layer

```
142    def __get_dynamodb_table(event, dynamodb):
143        if (is_pooled_deploy=='true'):
144            accesskey = event['requestContext']['authorizer']['accesskey']
145            secretkey = event['requestContext']['authorizer']['secretkey']
146            sessiontoken = event['requestContext']['authorizer']['sessiontoken']
147            dynamodb = boto3.resource('dynamodb',
148                    aws_access_key_id=accesskey,
149                    aws_secret_access_key=secretkey,
150                    aws_session_token=sessiontoken
151                    )
152        else:
153            if not dynamodb:
154                dynamodb = boto3.resource('dynamodb')
155
156        return dynamodb.Table(table_name)
157
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

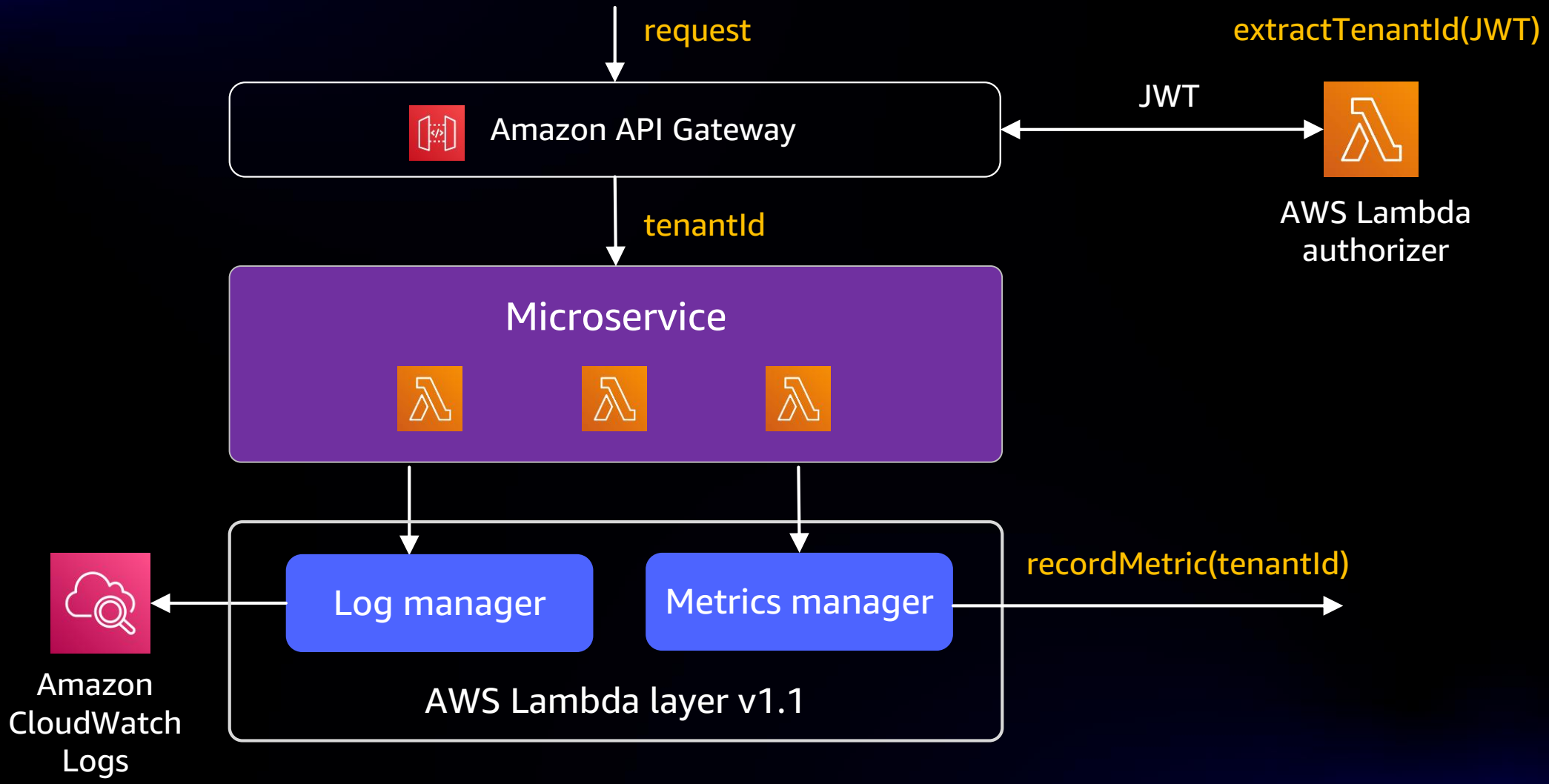
Pool-based partition with Amazon DynamoDB

- Tenant data in the same table
- TenantID used as ShardID
- Isolation policies applied by ShardID



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Inside application microservices



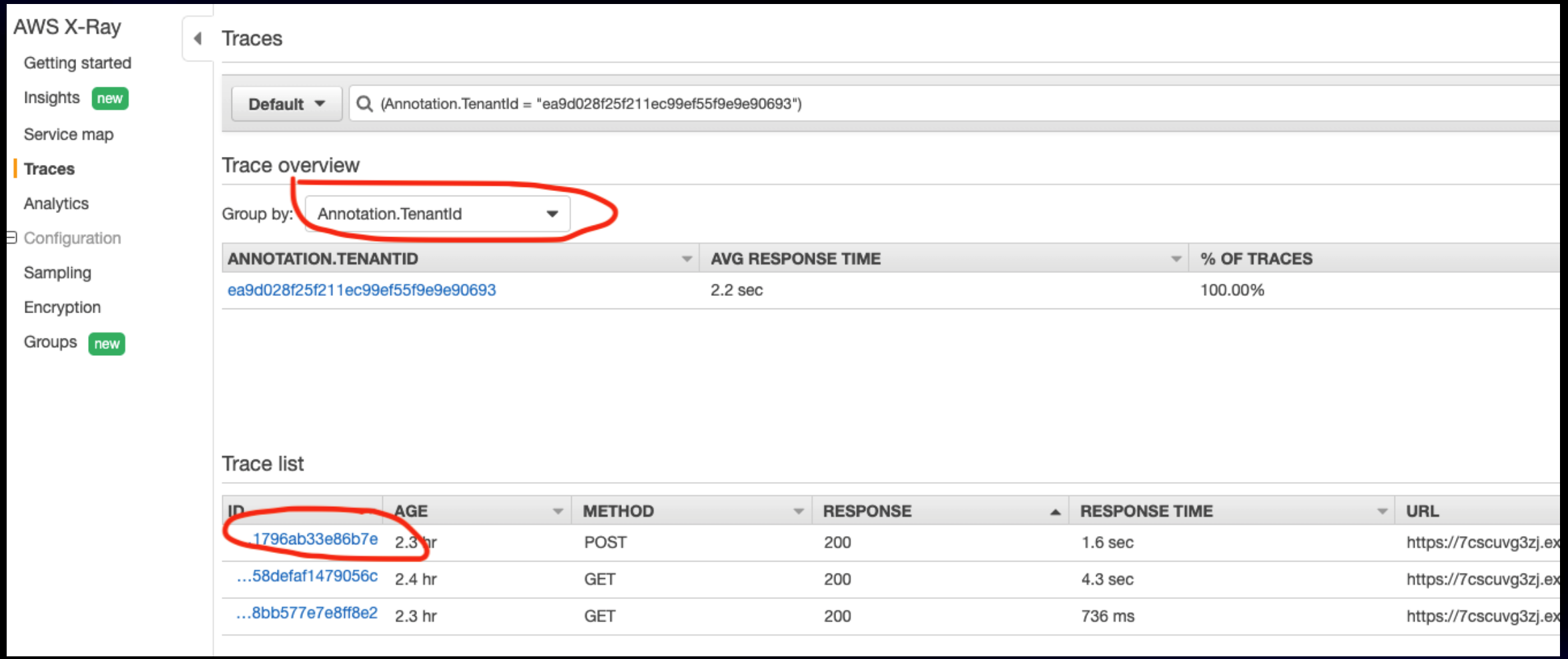
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Multi-tenant helper layers in action

```
14    def get_order(event, context):
15        tenantId = event['requestContext']['authorizer']['tenantId']
16        tracer.put_annotation(key="TenantId", value=tenantId)
17
18        logger.log_with_tenant_context(event, "Request received to get a order")
19        params = event['pathParameters']
20        key = params['id']
21        logger.log_with_tenant_context(event, params)
22        order = order_service_dal.get_order(event, key)
23
24        logger.log_with_tenant_context(event, "Request completed to get a order")
25        metrics_manager.record_metric(event, "SingleOrderRequested", "Count", 1)
26        return utils.generate_response(order)
27
```

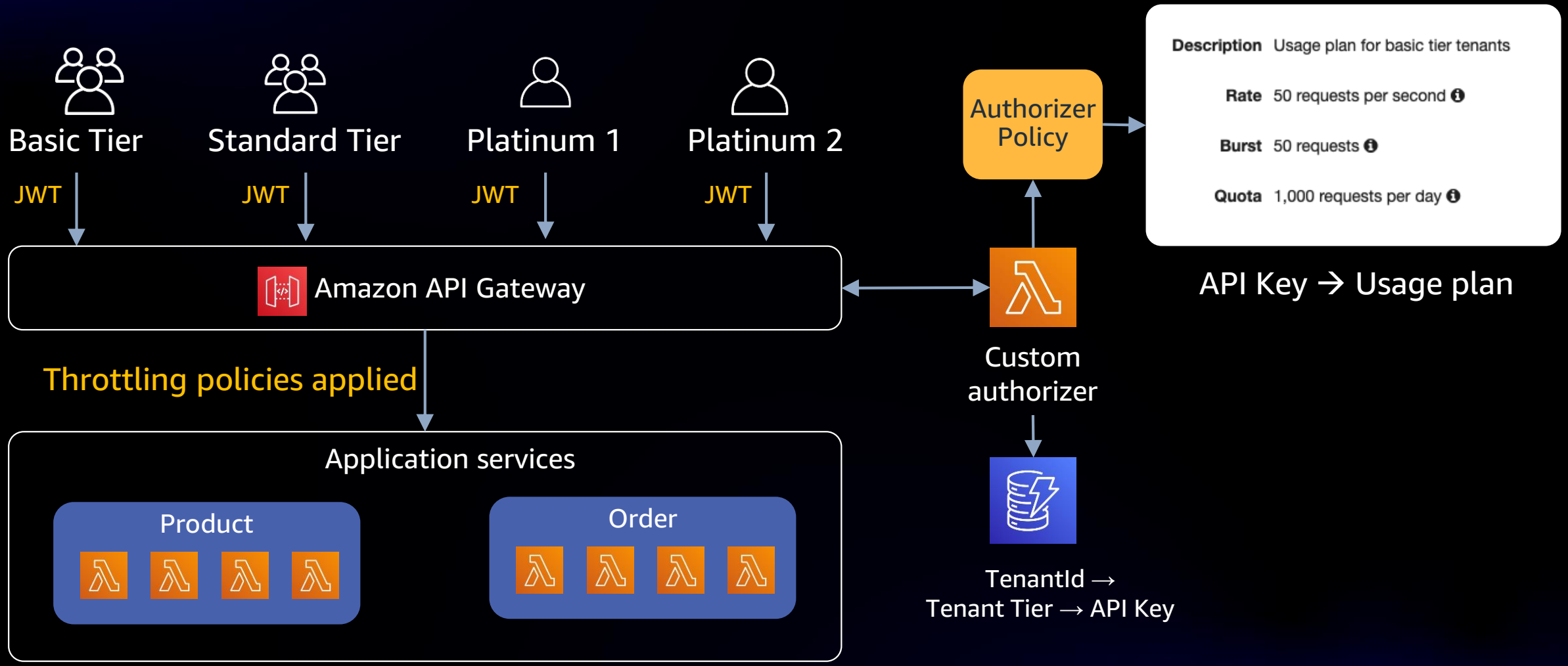
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

View activity by tenant



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Tier-based throttling policies



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Takeaways

- Serverless and SaaS are a good fit
- Find the approach that best addresses your isolation and tiering needs
- Embed all provisioning requirements in your onboarding automation
- Use API keys and usage plans to tier-based throttling
- Use AWS Lambda layers to provide shared multi-tenant constructs
- Use the serverless model to reduce the operational and development complexity of your SaaS environment
- Use the serverless SaaS reference solution to jumpstart your development

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Links to reference code and workshop

Serverless SaaS reference architecture

https://github.com/aws-samples/aws-saas-factory-ref-solution-serverless-saas

Serverless SaaS workshop

https://github.com/aws-samples/aws-serverless-saas-workshop

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Additional SaaS sessions

Breakout sessions

ARC306 - SaaS architecture patterns: From concept to implementation
ARC402 - EKS SaaS deep dive: A multi-tenant EKS SaaS solution
GPS209 - How to grow your SaaS business and drive revenue with AWS Marketplace

Workshops

ARC403 - Hands-on serverless SaaS: Building a serverless SaaS solution on AWS
ARC404 - Hands-on EKS SaaS: Building a multi-tenant SaaS solution on AWS

Chalk talks

ARC320 - Monolith to SaaS: Discover the way
GPS311 - Building multi-tenant-aware SaaS microservices on AWS
OPN309 - SaaS Boost vNext: Enabling new patterns and extensibility
ARC401 - Serverless SaaS deep dive: Inside a multi-tenant serverless solution

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

Tod Golding

todg@amazon.com



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.