# AWS re:Invent

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

ARC401-R2

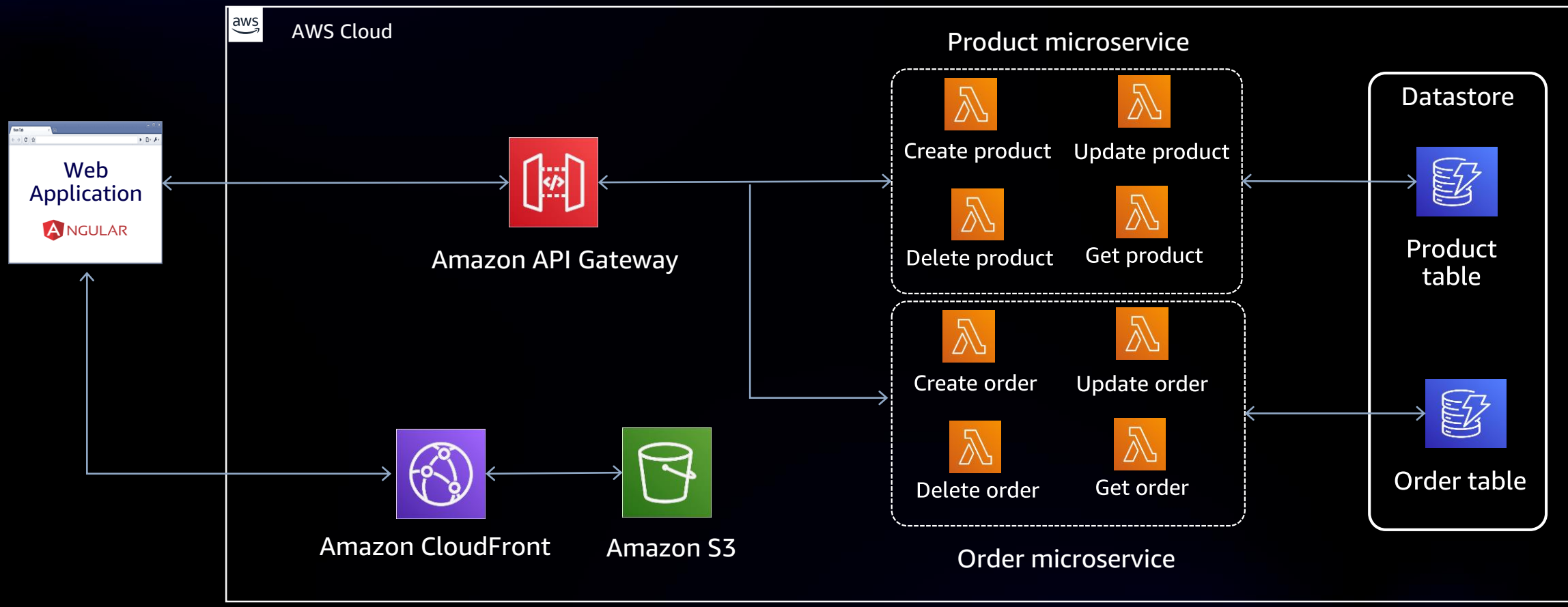
# Serverless SaaS deep dive: Inside a multi-tenant serverless solution


Anubhav Sharma (he/him)
Sr. Solutions Architect
AWS SaaS Factory

Tod Golding (he/him)
Principal Solutions Architect
AWS SaaS Factory

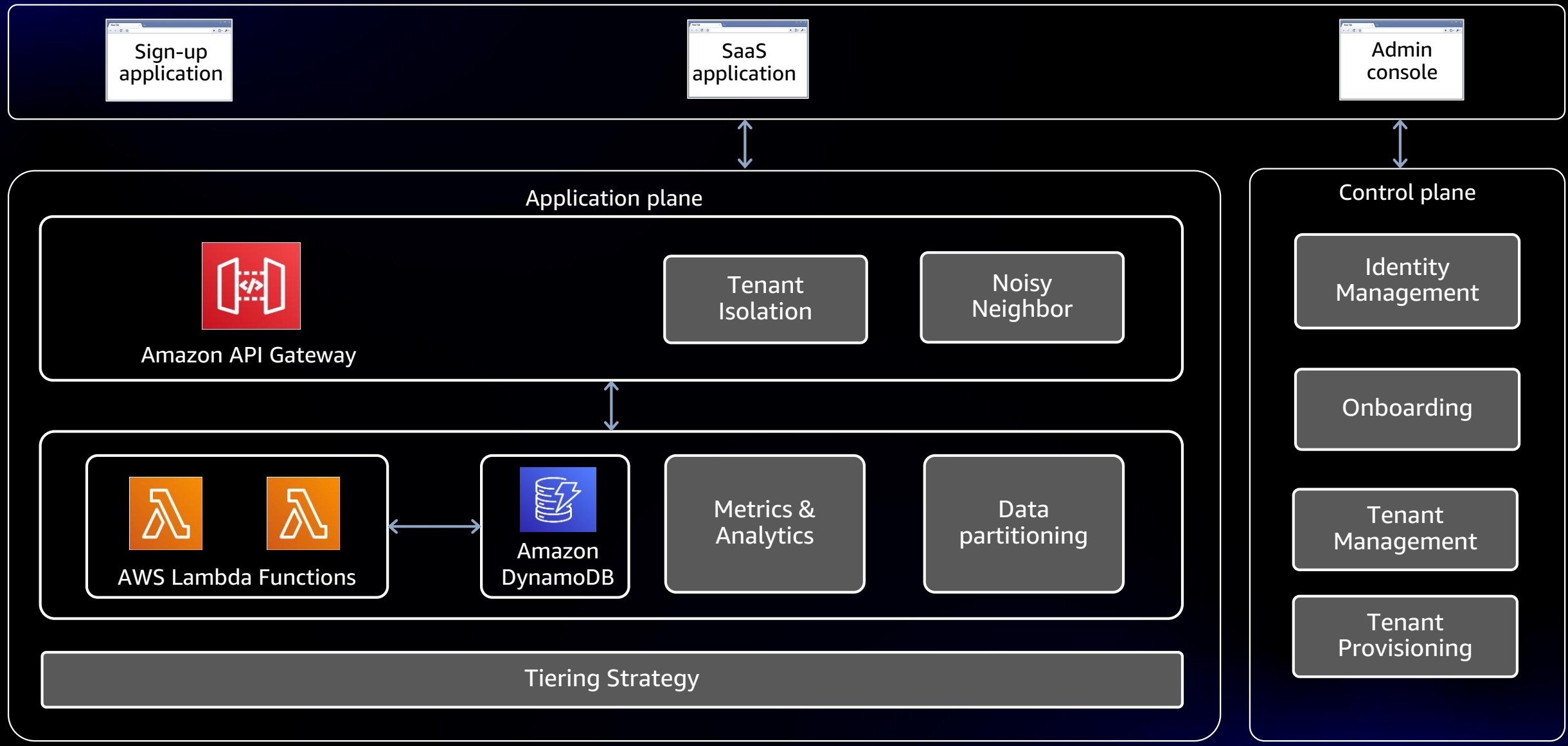

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

# A basic serverless web application



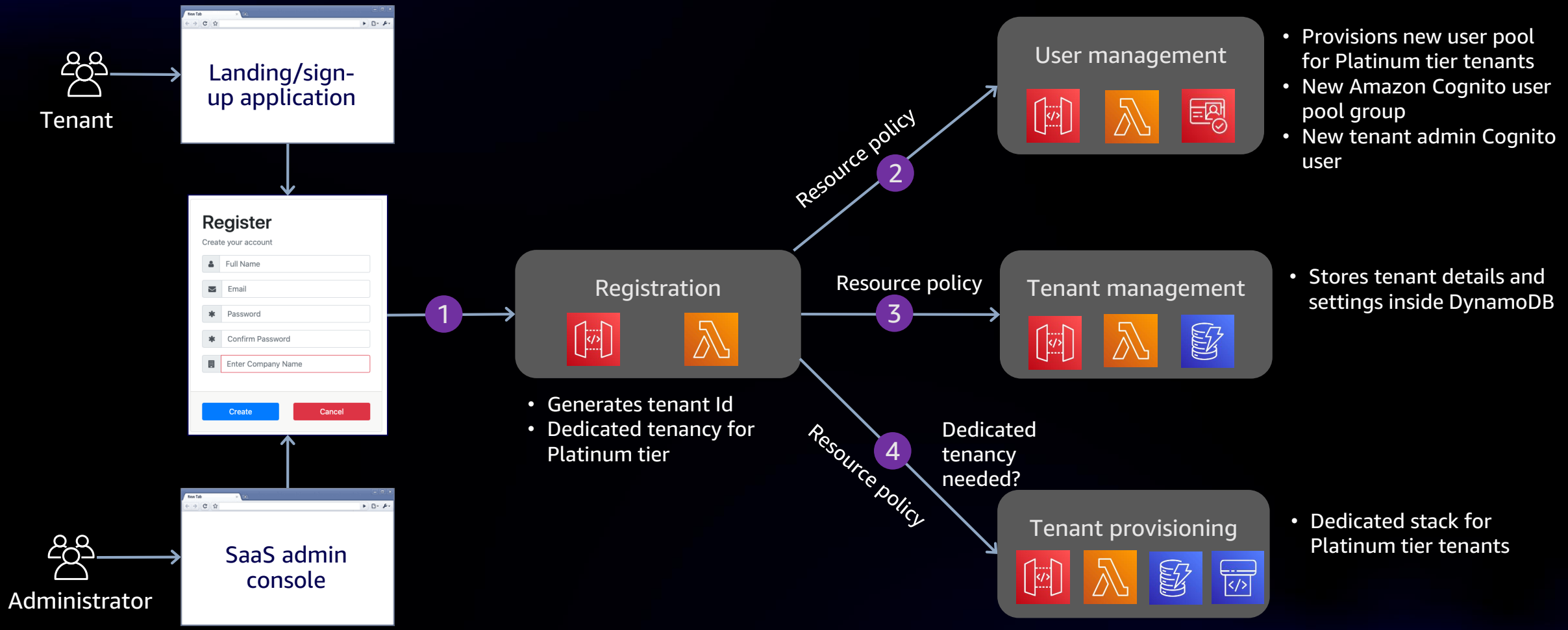

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

# Reference architecture – High-level view




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

# Tenant registration




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

# Tenant registration

```python
def register_tenant(event, context):
    try:
        tenant_id = uuid.uuid1().hex
        tenant_details['dedicatedTenancy'] = 'false'

        if (tenant_details['tenantTier'].upper() == utils.TenantTier.PLATINUM.value.upper()):
            tenant_details['dedicatedTenancy'] = 'true'
            api_key = platinum_tier_api_key
        elif (tenant_details['tenantTier'].upper() == utils.TenantTier.PREMIUM.value.upper()):
            api_key = premium_tier_api_key
        elif (tenant_details['tenantTier'].upper() == utils.TenantTier.STANDARD.value.upper()):
            api_key = standard_tier_api_key
        elif (tenant_details['tenantTier'].upper() == utils.TenantTier.BASIC.value.upper()):
            api_key = basic_tier_api_key

        tenant_details['tenantId'] = tenant_id
        tenant_details['apiKey'] = api_key

        auth = utils.get_auth(host, region)

        create_user_response = __create_tenant_admin_user(tenant_details, headers, auth, host, stage_name)

        create_tenant_response = __create_tenant(tenant_details, headers, auth, host, stage_name)

        if (tenant_details['dedicatedTenancy'].upper() == 'TRUE'):
            provision_tenant_response = __provision_tenant(tenant_details, headers, auth, host, stage_name)
            logger.info(provision_tenant_response)
```

```
- Effect: Deny
Principal: "*"
Action: "execute-api:Invoke"
Resource:
- !Join [
    "",
    [
      "execute-api:/", !Ref StageName, "/POST/user/tenant-admin"
    ]
  ]
Condition:
  StringNotEquals:
    aws:PrincipalArn:
      - !Ref RegisterTenantLambdaExecutionRoleArn
```

```python
def provision_tenant(event, context):

    tenant_details = json.loads(event['body'])

    try:
        response_ddb = table_tenant_stack_mapping.put_item(
            Item={
                    'tenantId': tenant_details['tenantId'],
                    'stackName': stack_name.format(tenant_details['tenantId']),
                    'applyLatestRelease': True,
                    'codeCommitId': ''
                }
            )

        logger.info(response_ddb)

        response_codepipeline = codepipeline.start_pipeline_execution(
            name='serverless-saas-pipeline'
        )

        logger.info(response_ddb)
```

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

# Tenant registration

```python
def create_user(event, context):
    user_details = json.loads(event['body'])

    logger.info("Request received to create new user")
    logger.info(event)

    tenant_id = user_details['tenantId']

    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': tenant_id
            }
        ]
```

Create user

```python
def create_tenant(event, context):

    try:
        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
                }
            )
```

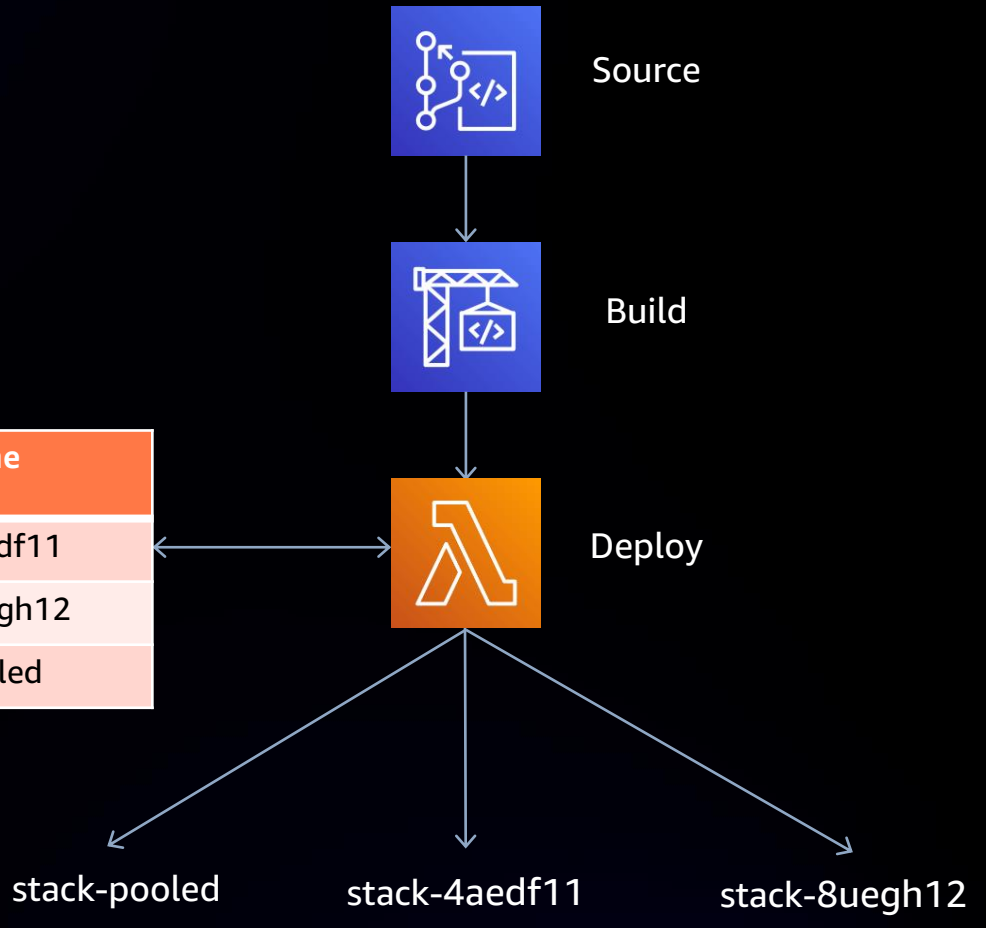
Create tenant

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

# Tenant CodePipeline

TenantStackMapping

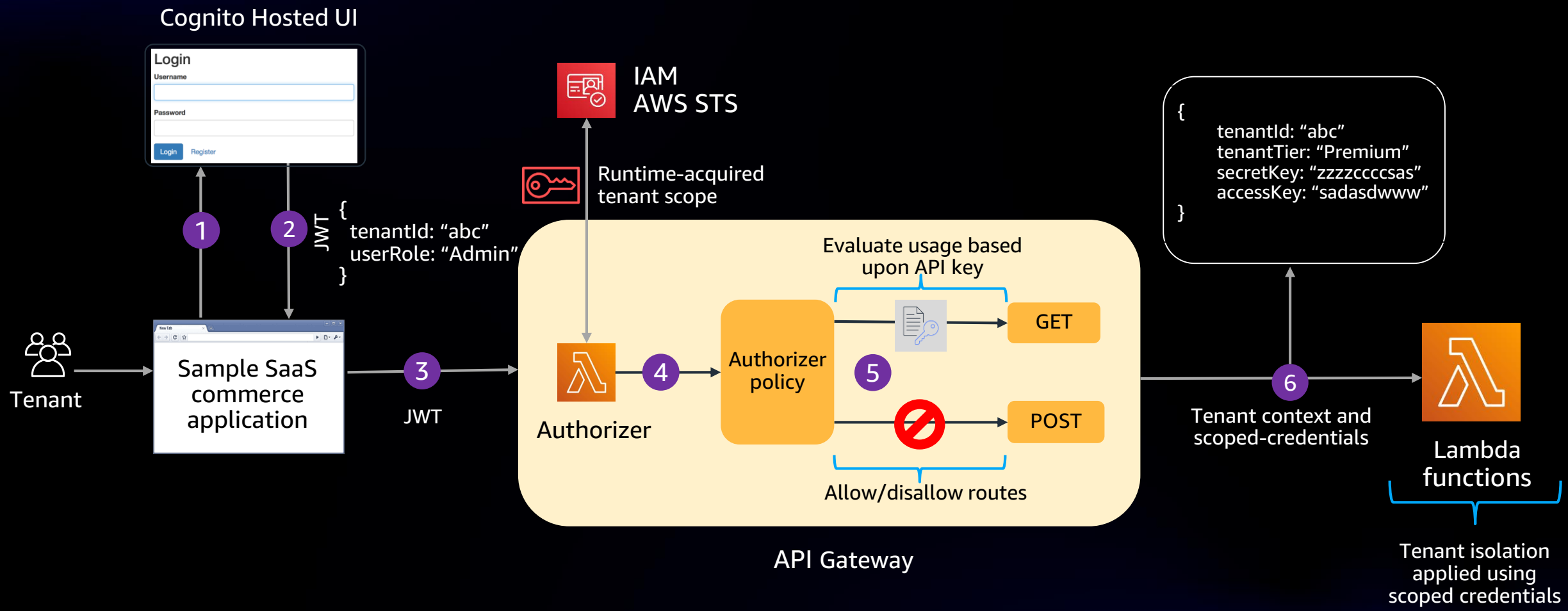
| tenantId | applyLatestRelease | codeCommitId | stackName |
|---|---|---|---|
| 4aedf11 | true | 0a65d | stack-4aedf11 |
| 8uegh12 | true | 0a65d | stack-8uegh12 |
| pooled | true | 0a65d | stack-pooled |




https://aws.amazon.com/blogs/devops/parallel-and-dynamic-saas-deployments-with-cdk-pipelines/

© 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.

# Authorizer output

```
#only tenant admin and system admin can do certain actions like create and disable users
if (auth_manager.isTenantAdmin(user_role) or auth_manager.isSystemAdmin(user_role)):
    policy.allowAllMethods()
    if (auth_manager.isTenantAdmin(user_role)):
        policy.denyMethod(HttpVerb.POST, "tenant-activation")
        policy.denyMethod(HttpVerb.GET, "tenants")
else:
    #if not tenant admin or system admin then only allow to get info and update info
    policy.allowMethod(HttpVerb.GET, "user/*")
    policy.allowMethod(HttpVerb.PUT, "user/*")


authResponse = policy.build()
```

```
#pass sts credentials to lambda
context = {
    'accesskey': credentials['AccessKeyId'], # $context.authorizer.key -> value
    'secretkey' : credentials['SecretAccessKey'],
    'sessiontoken' : credentials["SessionToken"],
    'userName': user_name,
    'tenantId': tenant_id,
    'userPoolId': userpool_id,
    'apiKey': api_key,
    'userRole': user_role
}

authResponse['context'] = context
authResponse['usageIdentifierKey'] = api_key

return authResponse
```

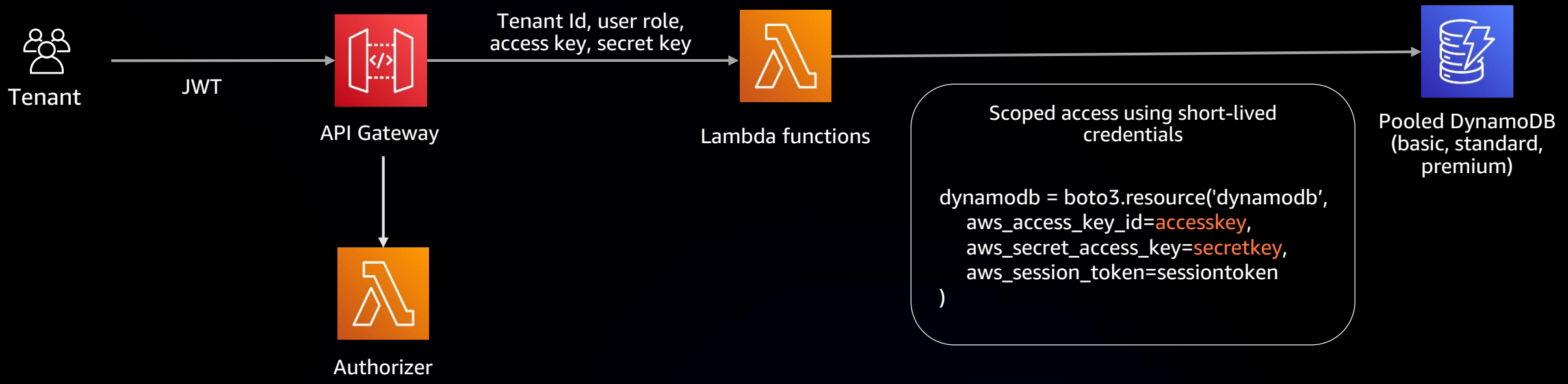
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

# Tenant isolation: Silo model




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

# Tenant isolation: Pooled model




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

# Dynamic policy

```
{
   "Effect": "Allow",
   "Action": [
      "dynamodb:UpdateItem",
      "dynamodb:GetItem",
      "dynamodb:PutItem",
      "dynamodb:DeleteItem",
      "dynamodb:Query"
   ],
   "Resource": [
      "arn:aws:dynamodb:{0}:{1}:table/Product-*".format(region, aws_account_id),
   ],
   "Condition": {
      "ForAllValues:StringLike": {
         "dynamodb:LeadingKeys": [
            "{0}-*".format(tenant_id)
         ]
      }
   }
}
```

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

# Code snippet: STS credentials

```
#get IAM policy
iam_policy = getPolicyForUser(user_role, tenant_id)

#use STS client to generate the credentials
assumed_role = sts_client.assume_role(
                    RoleArn=role_arn,
                    RoleSessionName="tenant-aware-session",
                    Policy=iam_policy,
)
credentials = assumed_role["Credentials"]

#pass sts credentials to lambda
context = {
                    'accesskey': credentials['AccessKeyId'], # $context.authorizer.key -> value
                    'secretkey' : credentials['SecretAccessKey’],
                    'sessiontoken' : credentials["SessionToken"],
}
```

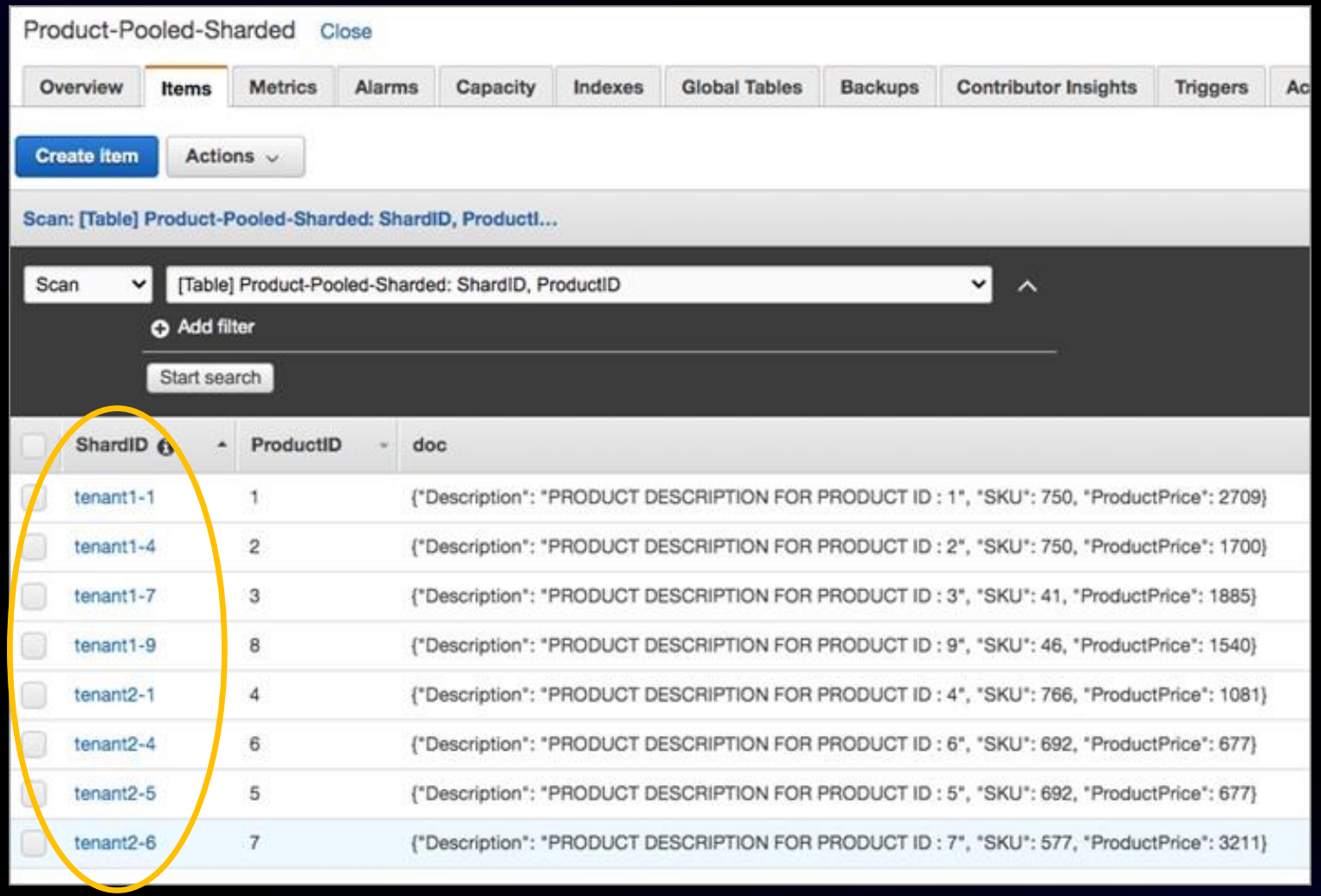
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

# 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.

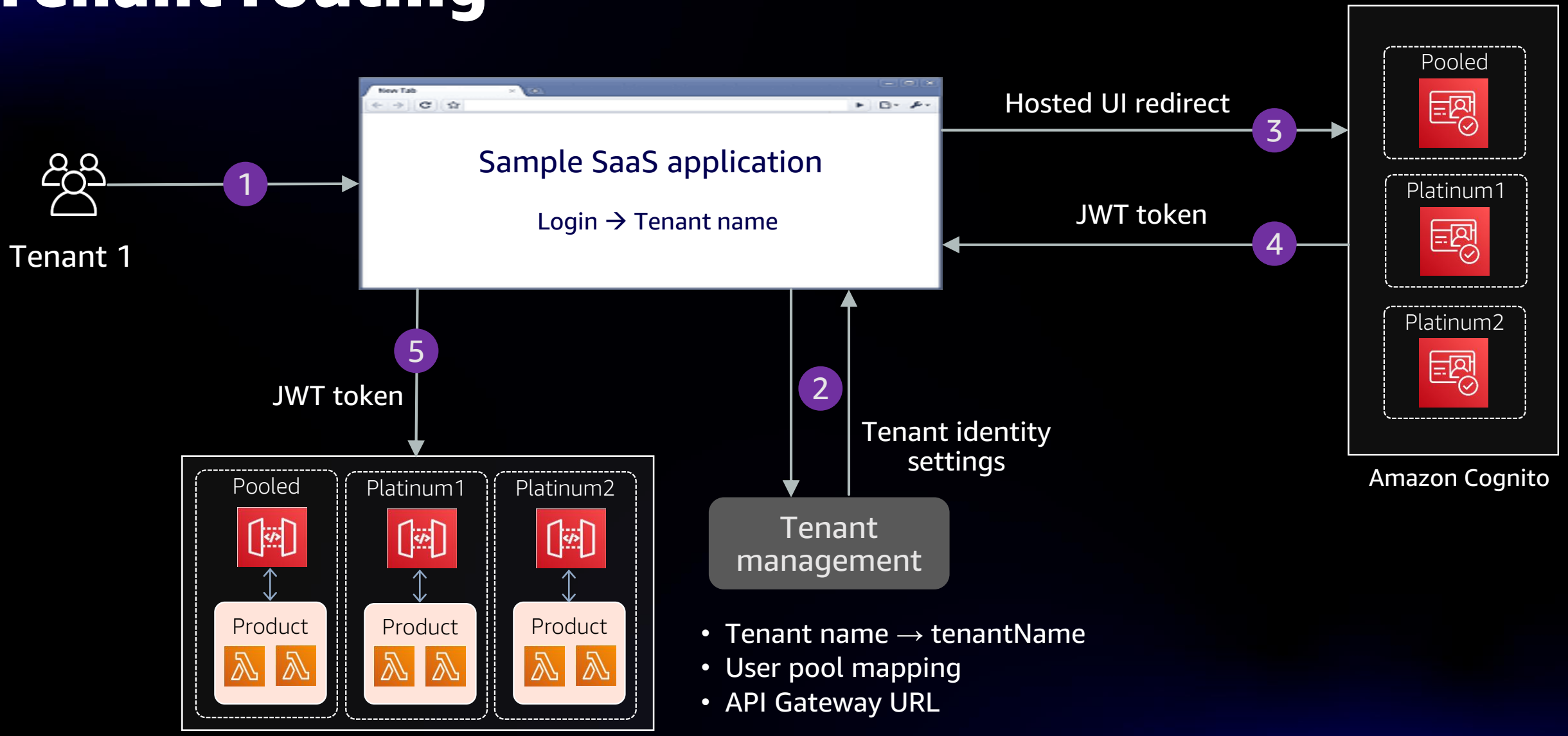
# DynamoDB partition key structure




https://aws.amazon.com/blogs/apn/partitioning-pooled-multi-tenant-saas-data-with-amazon-dynamodb/

© 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.

# 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.

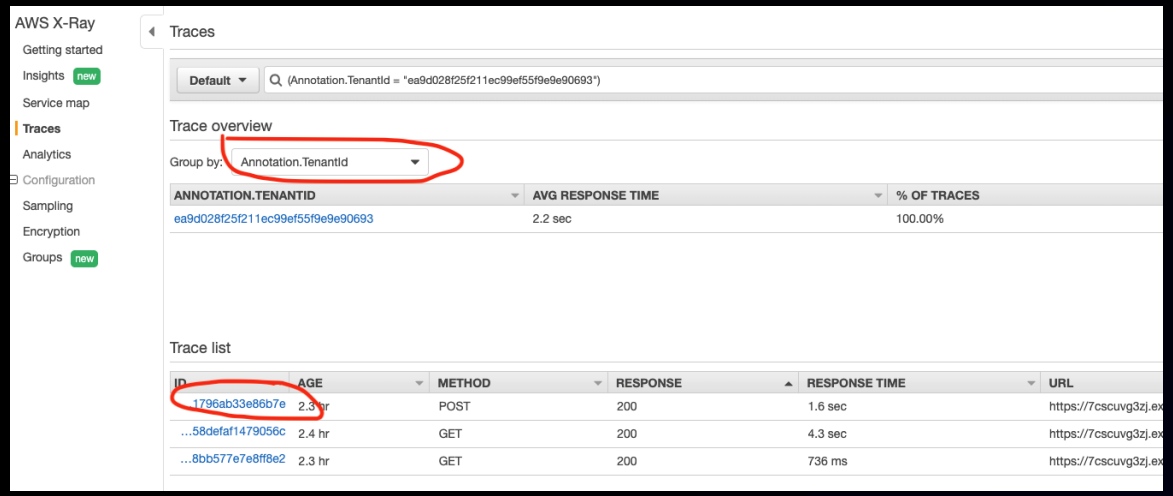
# Observability

## Logging with tenant context

```
2021-09-21T18:26:55.488-04:00    EXTENSION Name: cloudwatch_lambda_agent
2021-09-21T18:26:55.518-04:00    {"level":"INFO","location":"log_with_ten
```

```
{
    "level": "INFO",
    "location": "log_with_tenant_context:19",
    "message": "Request received to create a product",
    "timestamp": "2021-09-21 22:26:55,499+0000",
    "service": "ProductService",
    "tenant_id": "4fb565351b0a11ecab15250a6772c4a2",
    "xray_trace_id": "1-614a5c2e-76c4853c4e86383723b27a9d"
}
```

## Multi-tenant tracing with annotations




## Multi-tenant metrics (EMF)

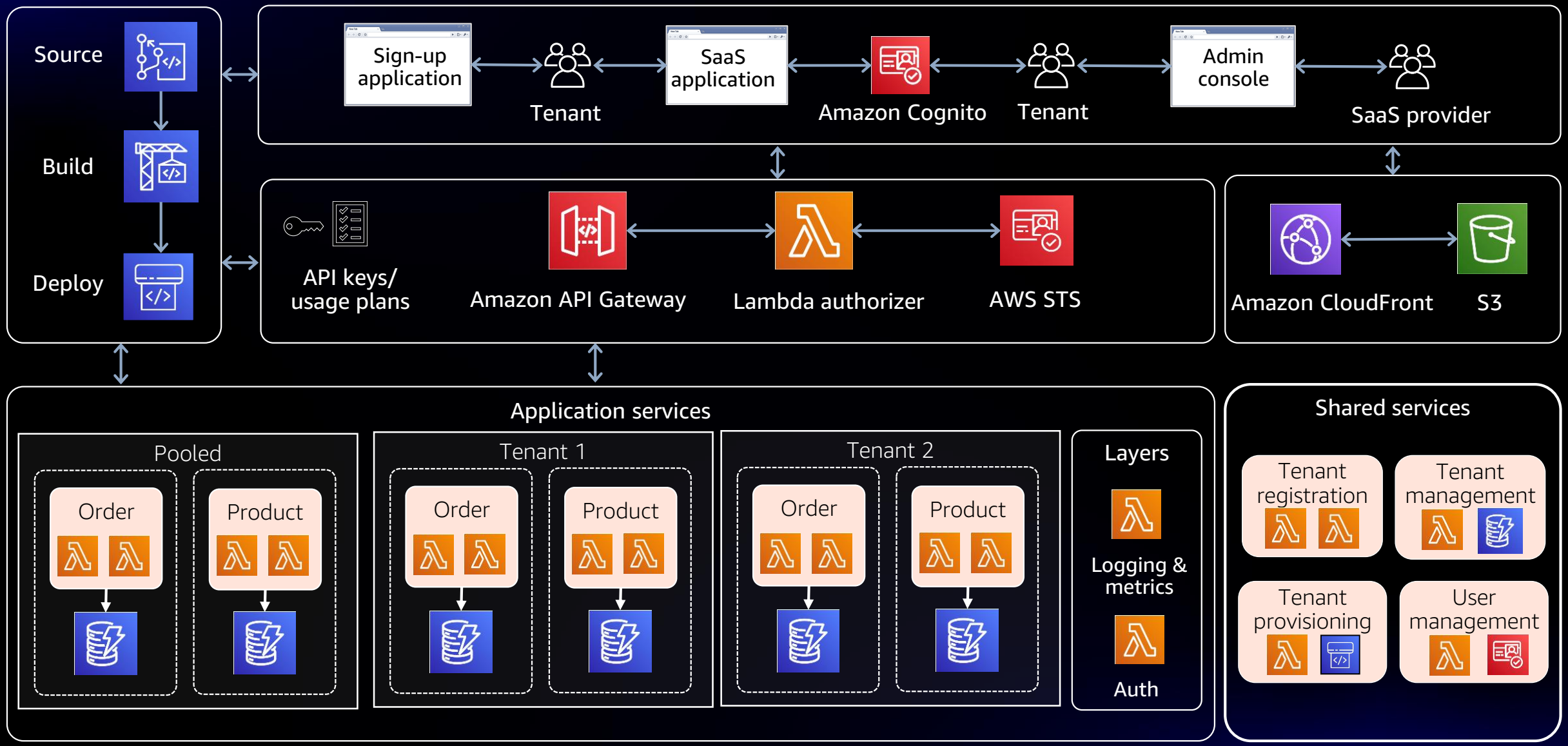
```
2021-08-24T17:39:07.102-04:00    {"_aws": {"Timestamp": 1629841147102, "CloudWatchMetrics": [{"Nam
```

```
{
    "_aws": {
        "Timestamp": 1629841147102,
        "CloudWatchMetrics": [
            {
                "Namespace": "ServerlessSaaS",
                "Dimensions": [
                    [
                        "tenant_id",
                        "service"
                    ]
                ],
                "Metrics": [
                    {
                        "Name": "ProductCreated",
                        "Unit": "Count"
                    }
                ]
            }
        ]
    },
    "tenant_id": "fc651ed9052211eca250e538bb53ea77",
    "service": "ProductService",
    "ProductCreated": [
        1
    ]
}
```

```
2021-08-24T17:39:07.239-04:00    END RequestId: c4fbc1b9-415a-4cba-aac8-2b8f26256ce6
```

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

# A SaaS based web application




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

# Reference solution

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

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

# Additional SaaS sessions

## Breakout Sessions

- ARC306 – SaaS architecture patterns: From concept to implementation
- ARC405 – Inside a working serverless SaaS reference solution
- ARC402 – Amazon 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 Amazon EKS SaaS: Building a multi-tenant SaaS solution on AWS

## Chalk Talks

- ARC320 – From Monolith to SaaS: Discover the way
- GPS311 – Building multi-tenant-aware SaaS microservices on AWS
- OPN309 – AWS 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!

Anubhav Sharma
shaanubh@amazon.com

Tod Golding
todg@amazon.com

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