AWS re:Invent

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

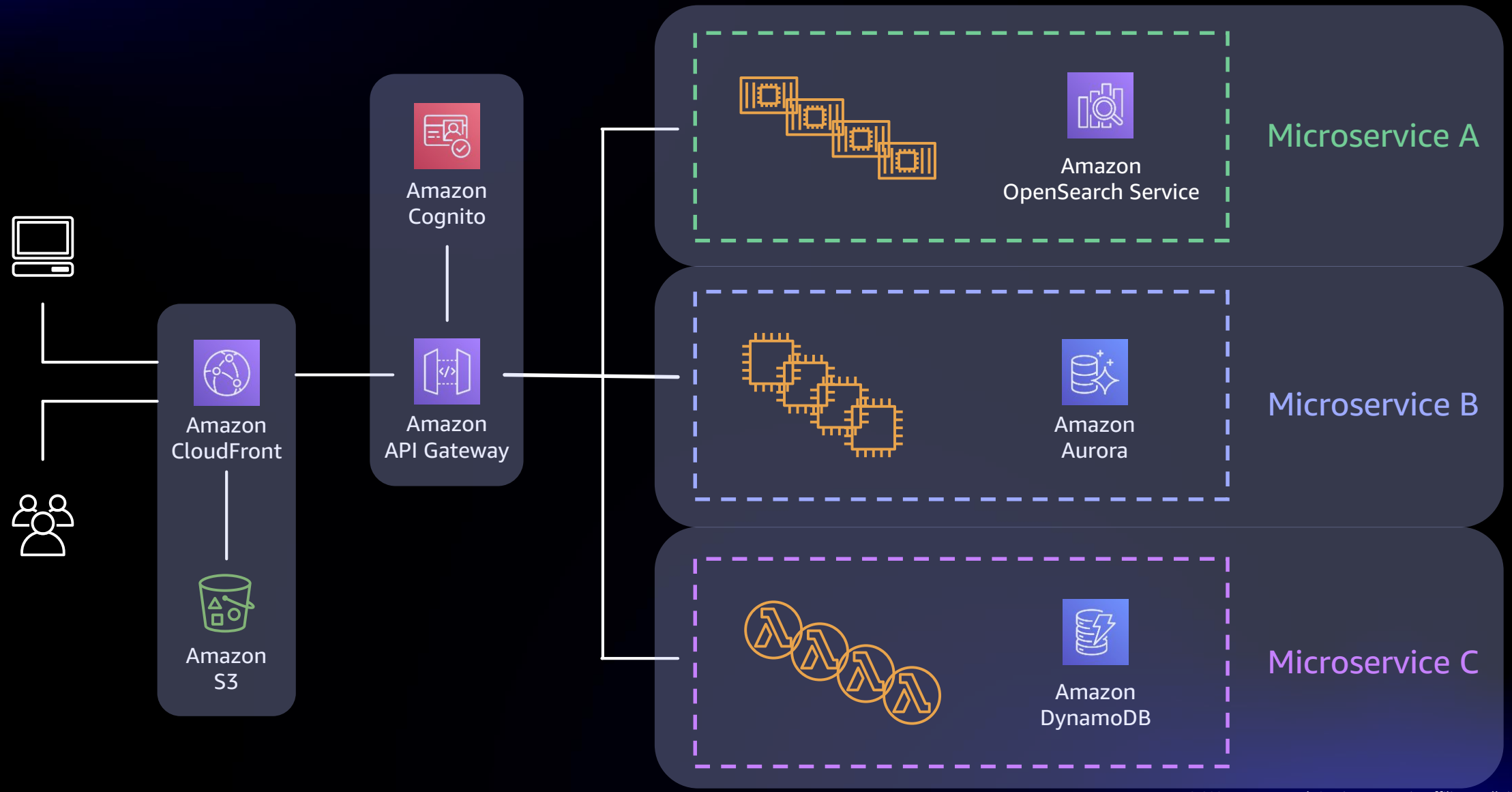
GPS311

Building multi-tenant-aware SaaS microservices on AWS

Michael Beardsley (he/him)
Partner Solutions Architect
AWS SaaS Factory

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

Typical microservices architecture



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

Multi-tenant awareness



What is tenant context? Why is it so important? Doesn’t this only matter when I’m sharing infrastructure?



How does this impact my application code? What are the implications of this overhead? How do I get it right?



Implementation tips, tricks, and tradeoffs. What are your ideas?

aws

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

Multi-tenant considerations



Identity



Logging



Metrics



Isolation

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

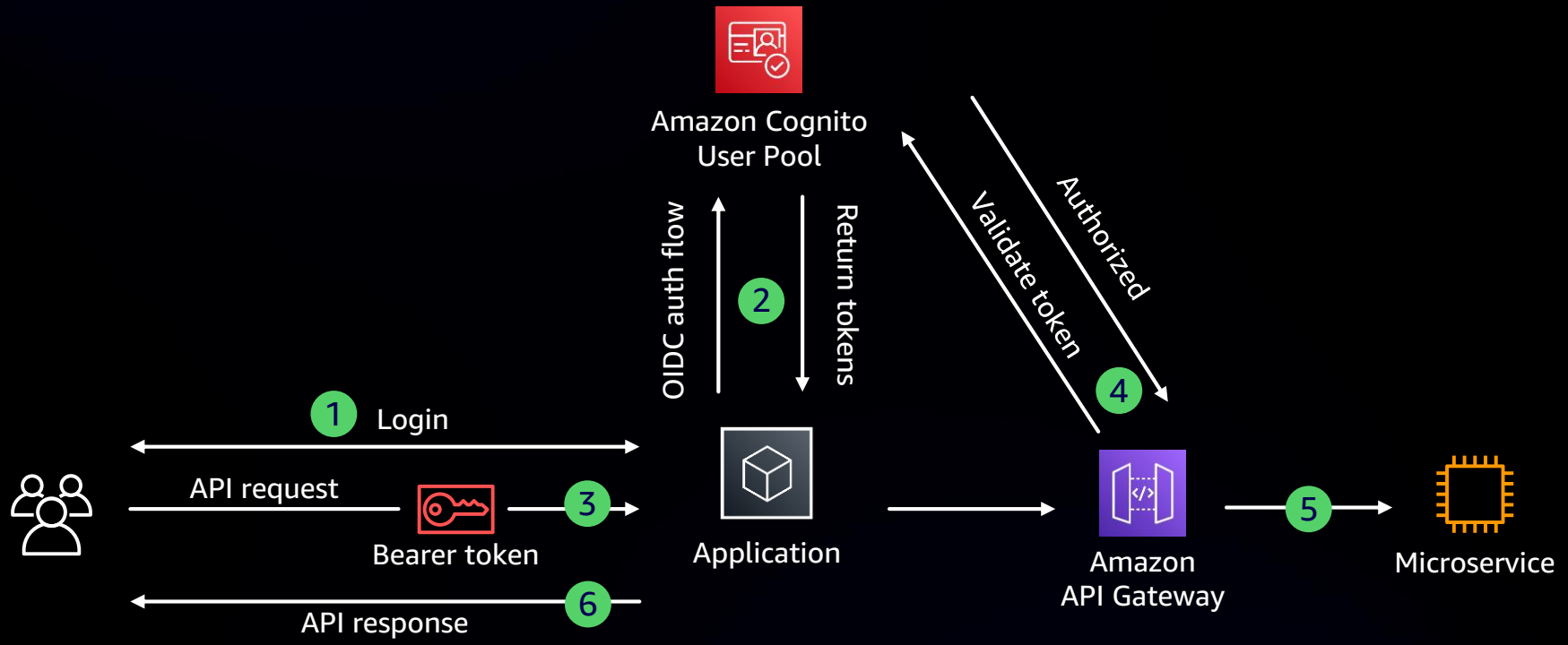
Code without tenant awareness

```
   @app.route('/orders/<user_id>')
1  def get_orders(user_id):
2    ddb = boto3.client('dynamodb')
3    scan = ddb.scan(
4      TableName = "orders",
5      FilterExpression = "user_id = :user_id",
6      ExpressionAttributeValues = {":user_id", {"S": user_id}}
7    )
8    orders = [from_attribute_value_map(order) for order in scan['Items']]
9    return orders
```

aws

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

Auth flow



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

Adding identity support

```
1  def get_orders(user_id):
2    auth = request.headers.get('Authorization')
3    bearer_token = re.split(r"^[B|b]earer +", auth)[1]
4    token = decode_token(bearer_token)
5    claims = get_claims(token)
6    tenant_id = claims['custom:tenant_id']
7
8    ddb = boto3.client('dynamodb')
9    query = ddb.query(
10     TableName = "orders",
11     KeyConditionExpression = "tenant_id = :tenant_id",
12     FilterExpression = "user_id = :user_id",
13     ExpressionAttributeValues = {":tenant_id", {"S": tenant_id}, ":user_id", {"S": user_id}}
14   )
15   orders = [from_attribute_value_map(order) for order in query['Items']]
16   return orders
```

aws

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

Adding tenant-aware logging

```
1  def get_orders(user_id):
2    tenant_id = token_helper.get_tenant_id(request)
3    ddb = boto3.client('dynamodb')
4    orders = []
5    try:
6      query = ddb.query(
7        TableName = "orders",
8        KeyConditionExpression = "tenant_id = :tenant_id",
9        FilterExpression = "user_id = :user_id",
10       ExpressionAttributeValues = {":tenant_id", {"S": tenant_id}, ":user_id", {"S": user_id}}
11     )
12     orders = [from_attribute_value_map(order) for order in query['Items']]
13   except exceptions.ClientError as ddb_error:
14     msg = {
15       "tenant_id": tenant_id, "user_id": user_id, "action": "get_orders", "message": str(ddb_error)
16     }
17     logger.error(json.dumps(msg))
18   return orders
```

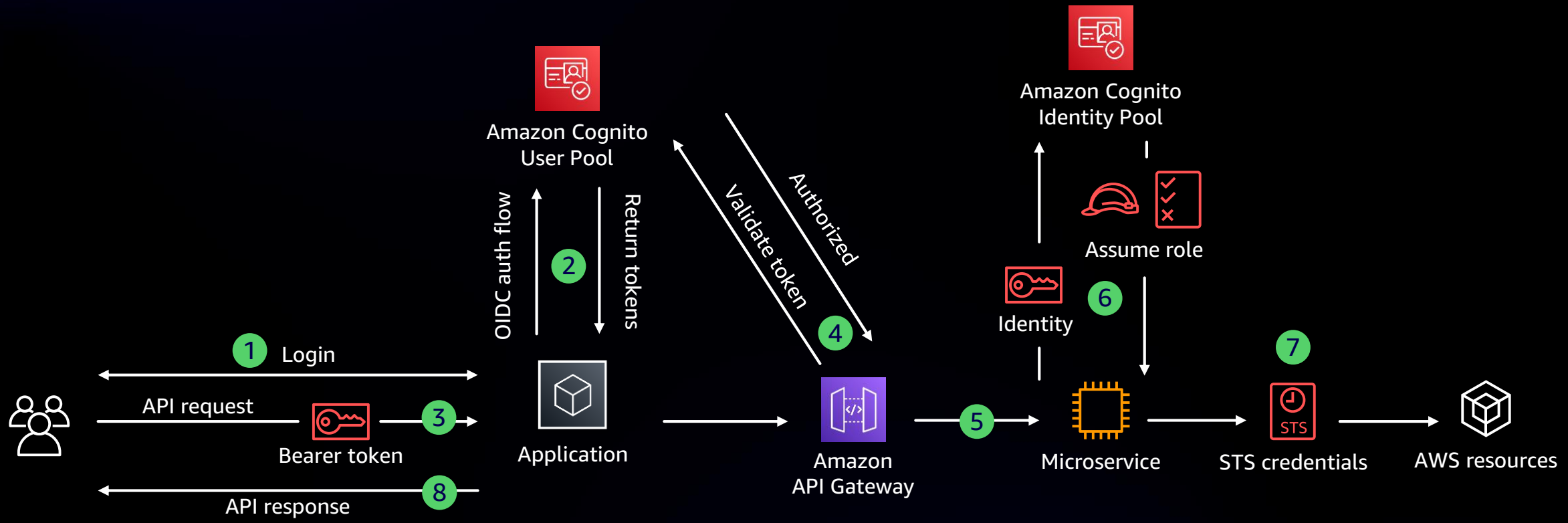
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Adding tenant-aware metrics

```
5   try:
6     query = ddb.query(
        ...
10      ExpressionAttributeValues = {":tenant_id", {"S": tenant_id}, ":user_id", {"S": user_id}},
11      ReturnConsumedCapacity = "TOTAL"
12    )
13    orders = [from_attribute_value_map(order) for order in query['Items']]
14    metric = {
15      "tenant_id": tenant_id, "user_id": user_id, "action": "get_orders",
16      ”records”: query['Count'], “capacity”: query['ConsumedCapacity']['CapacityUnits']
17    }
18    firehose = boto3.client('firehose')
19    firehose.put_record(
20      DeliveryStreamName = “metrics”,
21      Record = metric
22    )
23  except exceptions.ClientError as ddb_error:
24    logging_helper.log(request, ddb_error)
25  return orders
```

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

Isolation flow



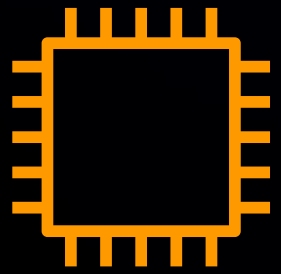
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Adding tenant isolation

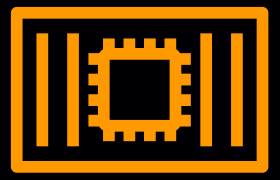
```
3   cognito = boto3.client('cognito-identity')
4   logins = {token_helper.get_provider_login(request): token_helper.get_bearer_token(request)}
5   cognito_identity = cognito.get_id(
6     IdentityPoolId = token_helper.get_identity_pool(request),
7     Logins = logins
8   )
9   tenant_credentials = cognito.get_credentials_for_identity(
10    IdentityId = cognito_identity['IdentityId'],
11    Logins = logins
12  )
13
14  tenant_scoped_session = boto3.Session(
15    aws_access_key_id = tenant_credentials['Credentials']['AccessKeyId'],
16    aws_secret_access_key = tenant_credentials['Credentials']['SecretAccessKey'],
17    aws_session_token = tenant_credentials['Credentials']['SessionToken']
18  )
19  ddb = tenant_scoped_session.client('dynamodb')
```

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

Challenges of sharing



Amazon EC2



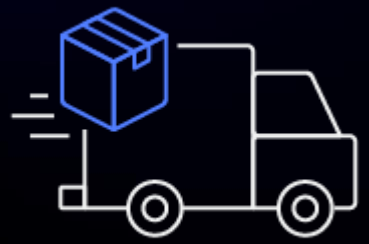
Containers



AWS Lambda

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

Shared libraries



Traditional packaging



Versioned



AWS Lambda layers



Same process



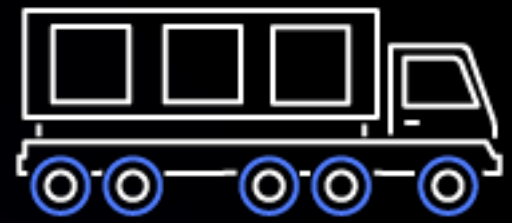
Common language

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

Background processes



Agent or daemon



Sidecar

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

Interceptors



Network proxies



Middle-ware



AWS Lambda extensions

© 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

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!

Michael Beardsley

mibeard@amazon.com



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