
AWS
re:Invent
NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV

ARC403-R

Amazon EKS SaaS deep dive: Inside a multi-tenant EKS solution

Tod Golding

Senior Principal Solutions Architect
AWS SaaS Factory

Toby Buckley

Senior Solutions Architect
AWS SaaS Factory

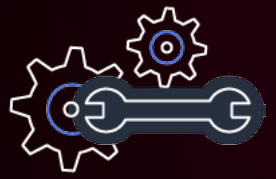
Amazon EKS: A natural fit for SaaS

Rapid scaling

Security constructs

Cost efficiency

Community tooling



Developer learning curve

Landscape of composable constructs

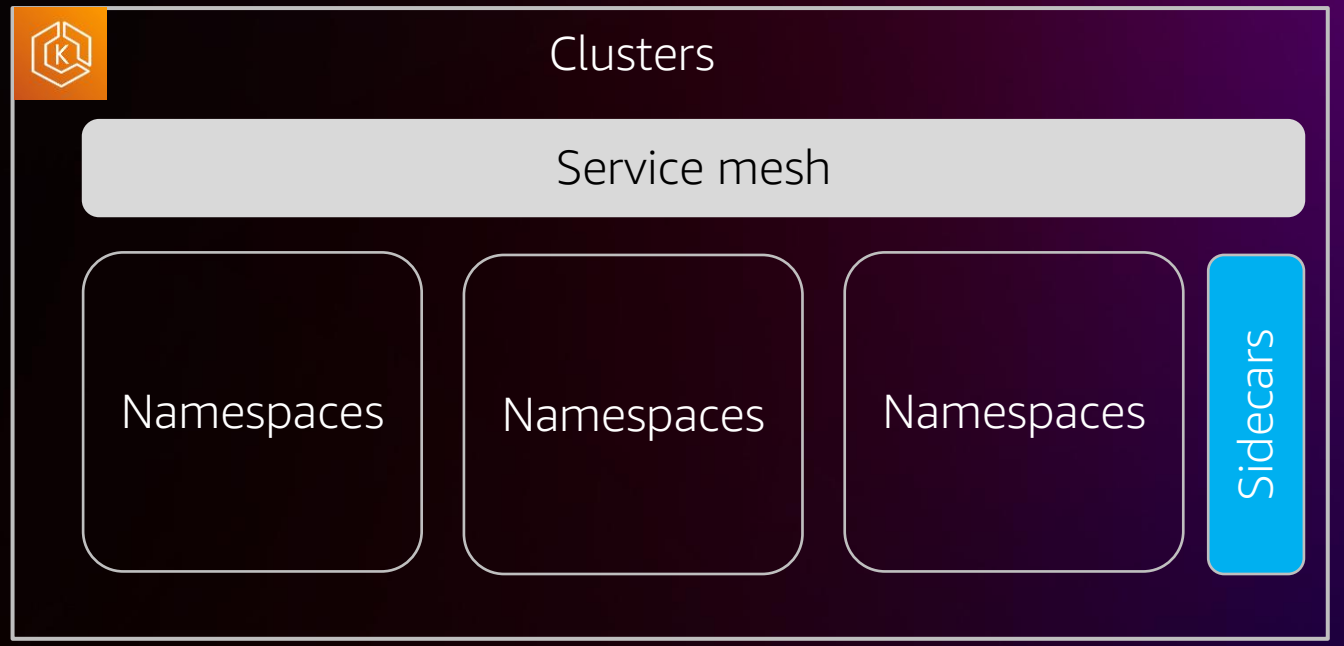
Rich provisioning and configuration

Helm

kubectl

cdk8s

Broad collection of architecture constructs



Let’s start with a deployment model

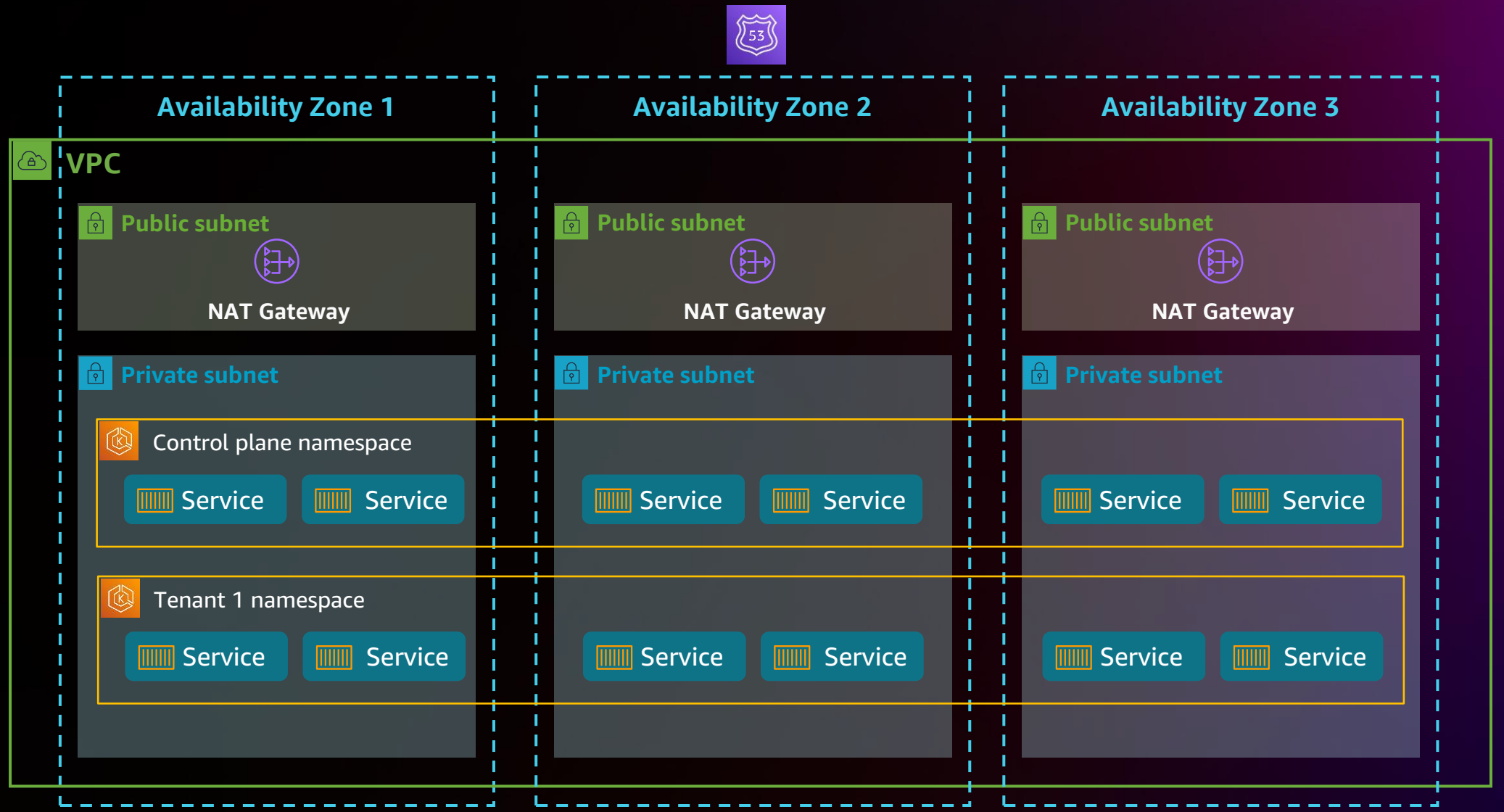
Application plane

- Which tiers/personas do we need to support?
- Are there compliance, domain, legacy considerations?
- What’s the isolation story of our solution?
- Are there key use cases where we might have bottlenecks?

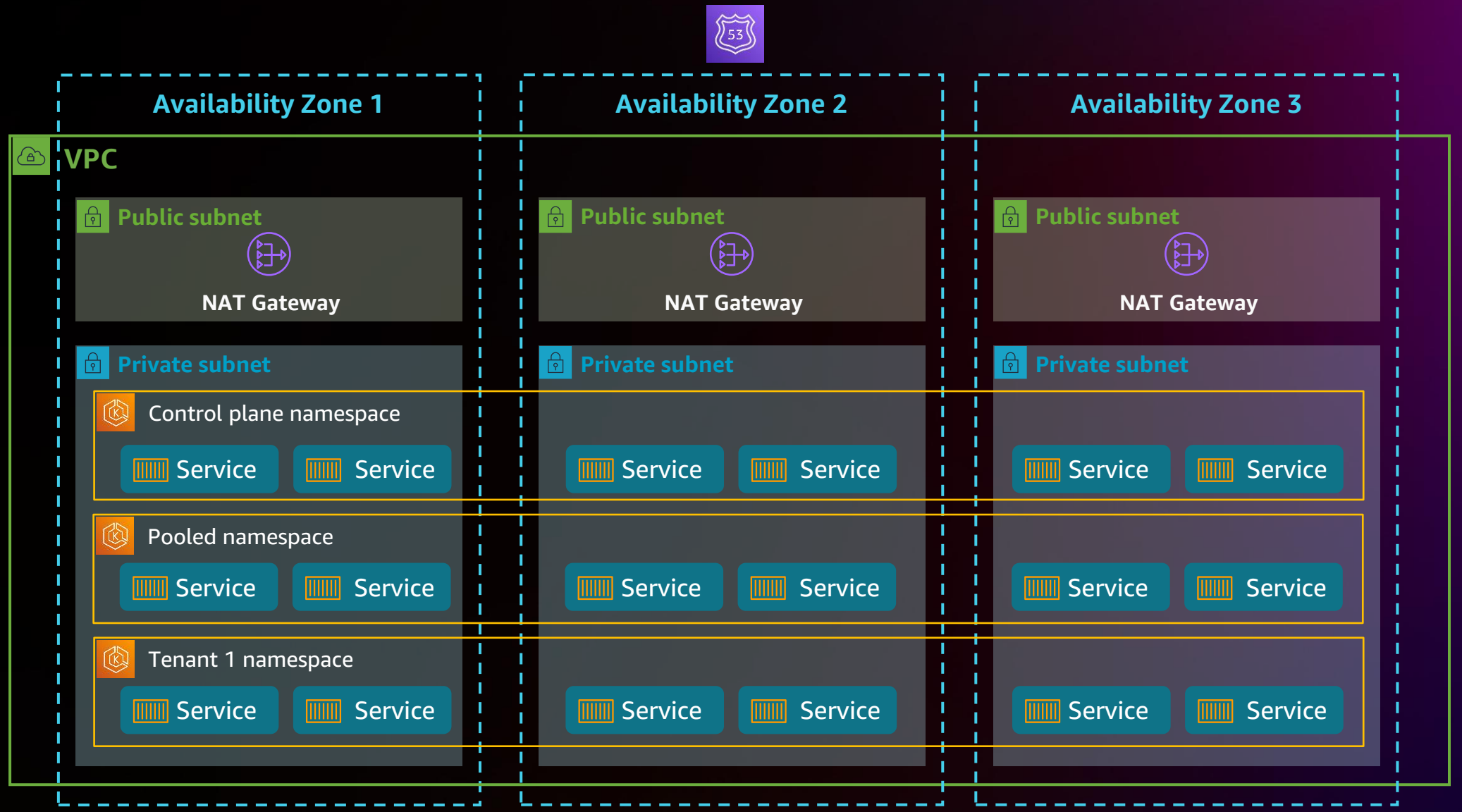
Control plane

- How do we plan to manage and administer our SaaS environment?
- What’s our identity model?
- How will we provision tenant environments?
- How will the control plane interact with the application plane?

Mapping these ideas to an Amazon EKS footprint

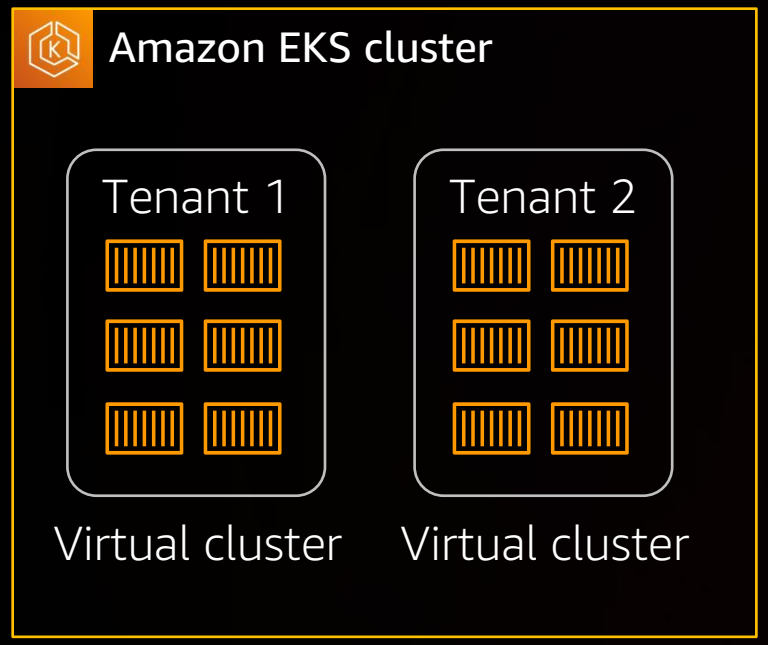


Mapping these ideas to an Amazon EKS footprint

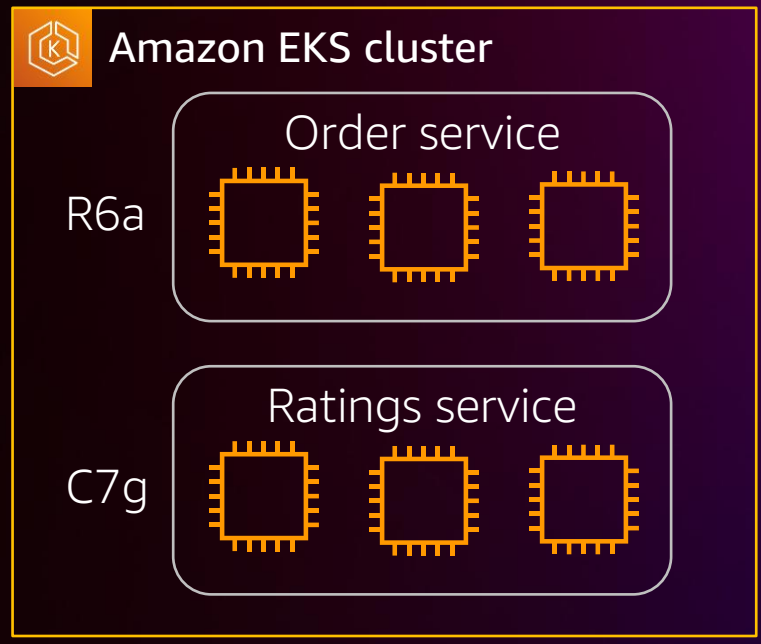


Additional deployment considerations

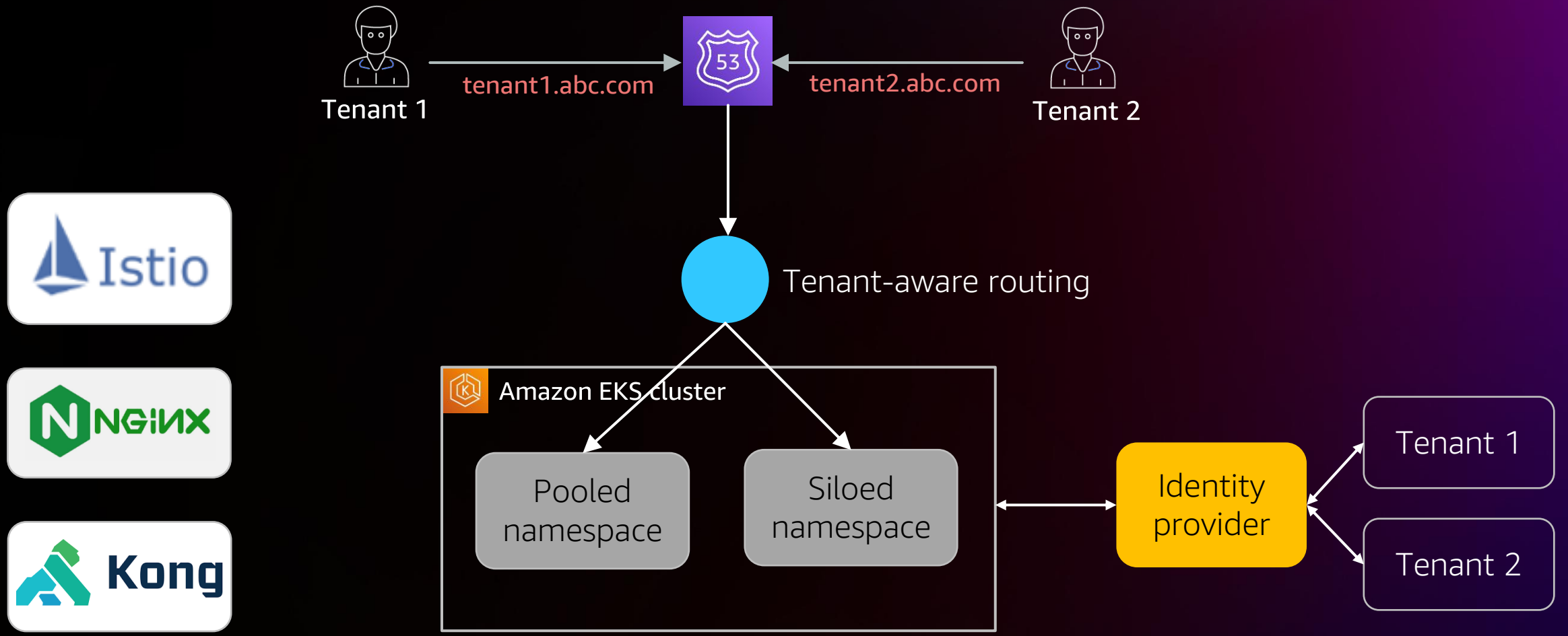
Siloed virtual clusters



Using node affinity



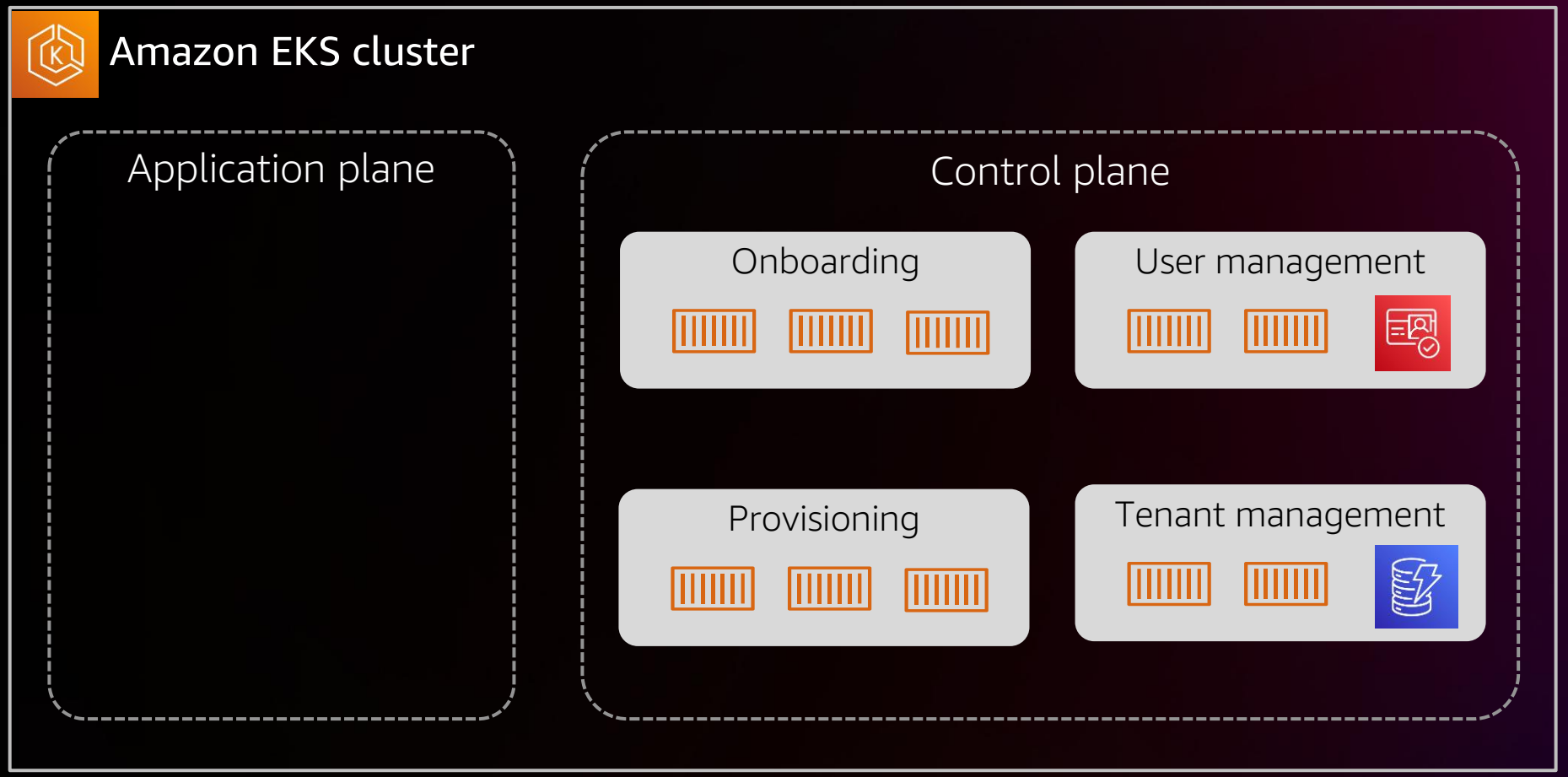
How will you identify tenants?



Provision your baseline environment

Create the control plane

Where's my K8S SDK?

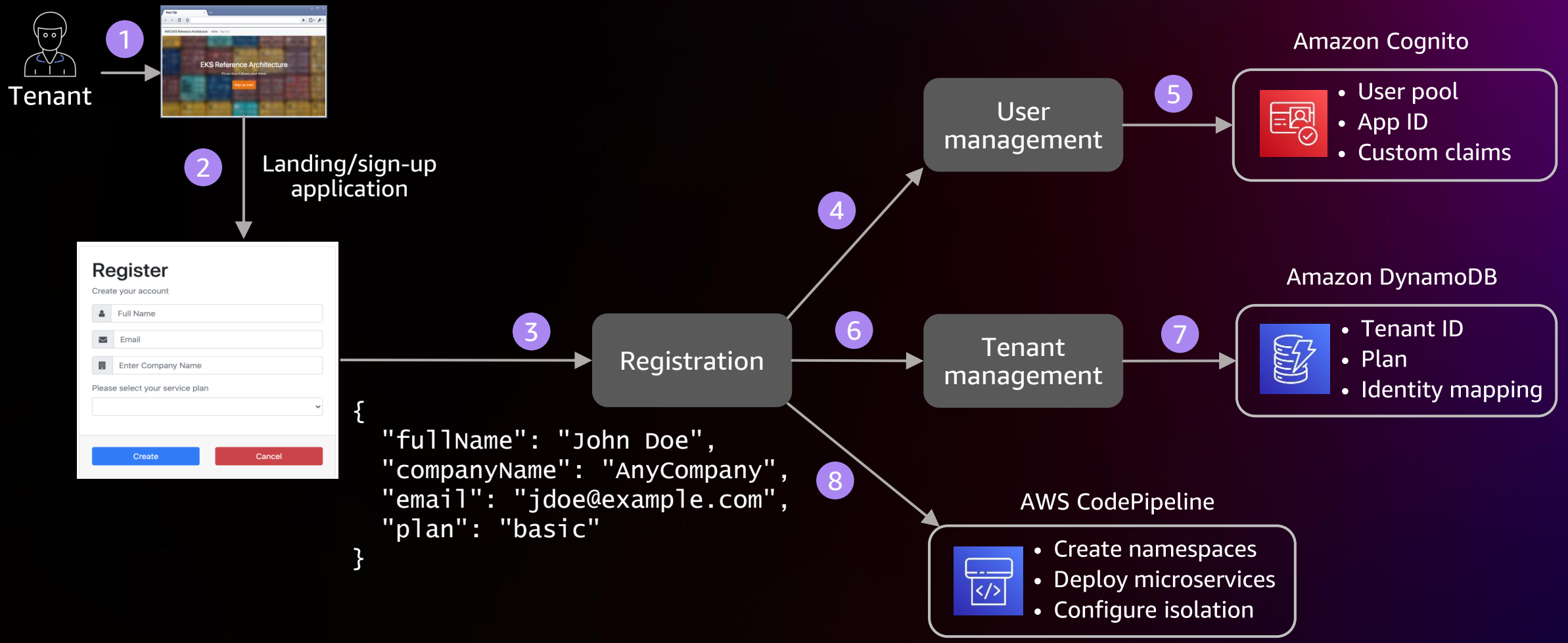


gitops

AWS CDK

AWS CodeBuild

Onboarding a new tenant



Inside the tenant registration service

```
77  public class TenantRegistrationService {
78    private static final String TENANT = "Tenant";
79    private static final Logger logger = LogManager.getLogger(TenantRegistrationService.class
80    private static final String SAAS_PROVIDER_METADATA = "SAAS_PROVIDER_METADATA";
81
82    public String registerTenant(TenantDetails tenant) {
83      String tenantId = null;
84
85      if (tenant != null) {
86        tenant = createTenant(tenant);
87        tenant = registerUser(tenant);
88        tenant = createTenantServices(tenant);
89
90        tenantId = tenant.getTenantId();
91        LoggingManager.logInfo(tenantId, "Tenant registration success!");
92
93        return tenantId;
94      } else {
95        logger.error("Error in tenant signup process. Please check the logs.");
96      }
97
98      return tenantId;
99    }
```

Creating the tenant

```
107  private TenantDetails createTenant(TenantDetails tenant) {
108    RestTemplate restTemplate = new RestTemplate();
109    String tenantManagementServiceUrl = "http://tenant-management-service/tenant/create";
110
111    ResponseEntity<TenantDetails> response = restTemplate.postForEntity(tenantManagementServiceUrl, tenant,
112        TenantDetails.class);
113
114    if (response != null)
115      logger.info("Tenant registration process is complete and a new tenant has been successfully created. =>"
116          + response.getBody().toString());
117    else
118      logger.error("Tenant registration process failure. Please check logs");
119
120    return response.getBody();
121  }
```

Creating tenant services

```
142     CreateStackRequest createRequest = new CreateStackRequest();
143     createRequest.setStackName(stackName);
144     createRequest.setTemplateURL(saaSProviderMetadata.getS3Endpoint());
145
146     List<Parameter> parameters = new ArrayList<Parameter>();
147     Parameter param = new Parameter();
148     param.setParameterKey("TenantName");
149     param.setParameterValue(tenant.getTenantId());
150     parameters.add(param);
151
152     Parameter customDomainParam = new Parameter();
153     customDomainParam.setParameterKey("CustomDomain");
154     customDomainParam.setParameterValue(tenant.getCustomDomain());
155     parameters.add(customDomainParam);
156
157     Parameter productServiceEcrRepoUriParam = new Parameter();
158     productServiceEcrRepoUriParam.setParameterKey("ProductServiceEcrRepoUri");
159     productServiceEcrRepoUriParam.setParameterValue(saaSProviderMetadata.getProductServiceEcrRepoUri());
160     parameters.add(productServiceEcrRepoUriParam);
161
162     Parameter orderServiceEcrRepoUriParam = new Parameter();
163     orderServiceEcrRepoUriParam.setParameterKey("OrderServiceEcrRepoUri");
164     orderServiceEcrRepoUriParam.setParameterValue(saaSProviderMetadata.getOrderServiceEcrRepoUri());
165     parameters.add(orderServiceEcrRepoUriParam);
166
167     createRequest.setParameters(parameters);
168
169     List<String> capabilities = new ArrayList<String>();
170     capabilities.add("CAPABILITY_IAM");
171     createRequest.setCapabilities(capabilities);
172     client.createStack(createRequest);
```

tenant_stack
(generated)

Tenant stack code build → buildspec.yaml

1 Update_tenant_provisioning_stack.sh → tenant_stack.yaml

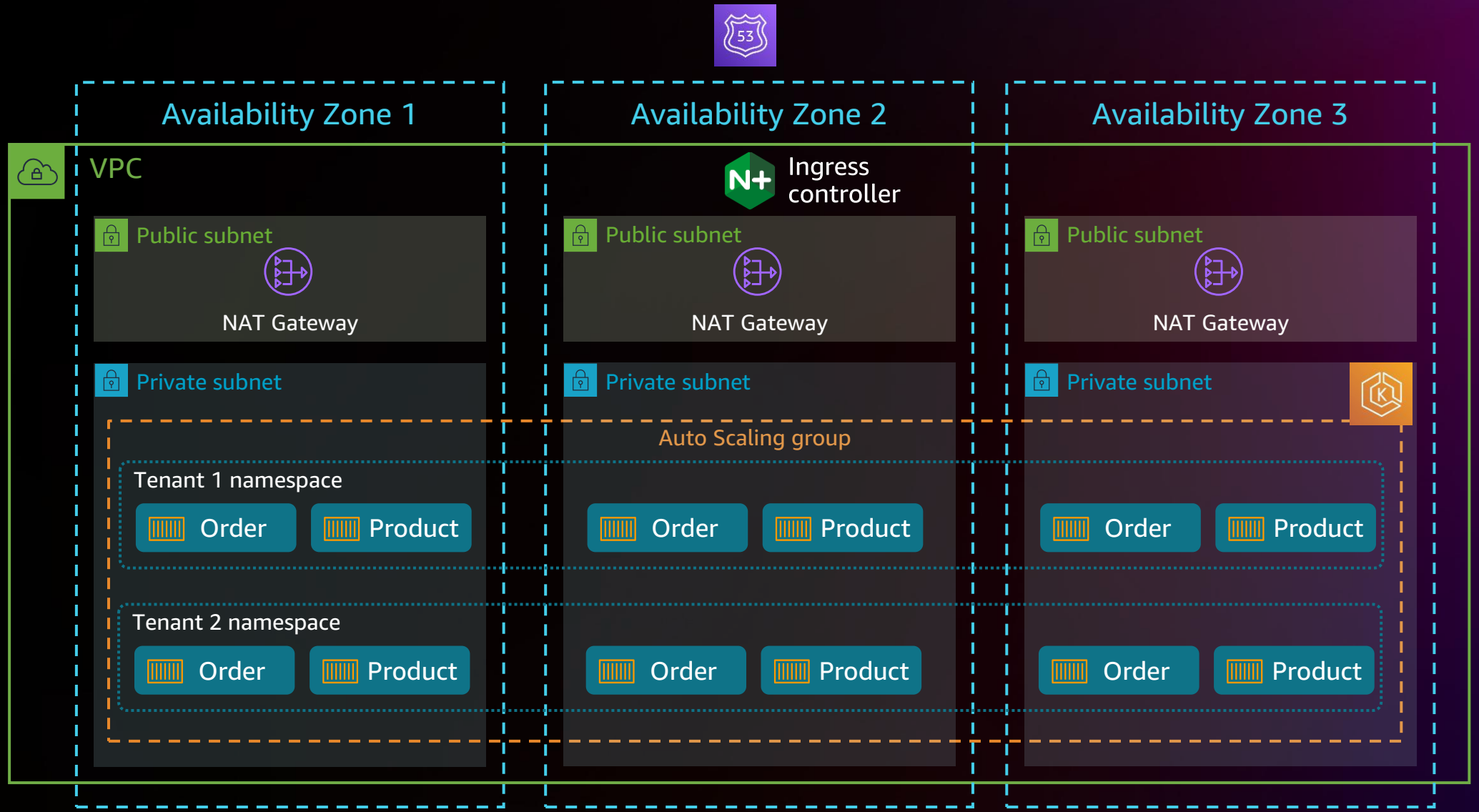
```
50  # Replace placeholders within the tenant stack template with values from our environment
51  sed 's,EKS_CLUSTER_PLACEHOLDER,'$EKS_CLUSTER_NAME',g' resources/templates/tenant-stack-master.yaml >
52      resources/templates/tenant-stack.yaml
53  sed -i 's,EKS_VPC_ID,'$EKS_VPC_ID',g' resources/templates/tenant-stack.yaml
54  sed -i 's,EKS_SUBNET_ID,'$EKS_SUBNET_ID',g' resources/templates/tenant-stack.yaml
55  sed -i 's,EKS_SECURITY_GROUP_ID,'$EKS_SECURITY_GROUP_ID',g' resources/templates/tenant-stack.yaml
```

2 Registration → createTenantServices() → tenant_stack.yaml → codeBuild

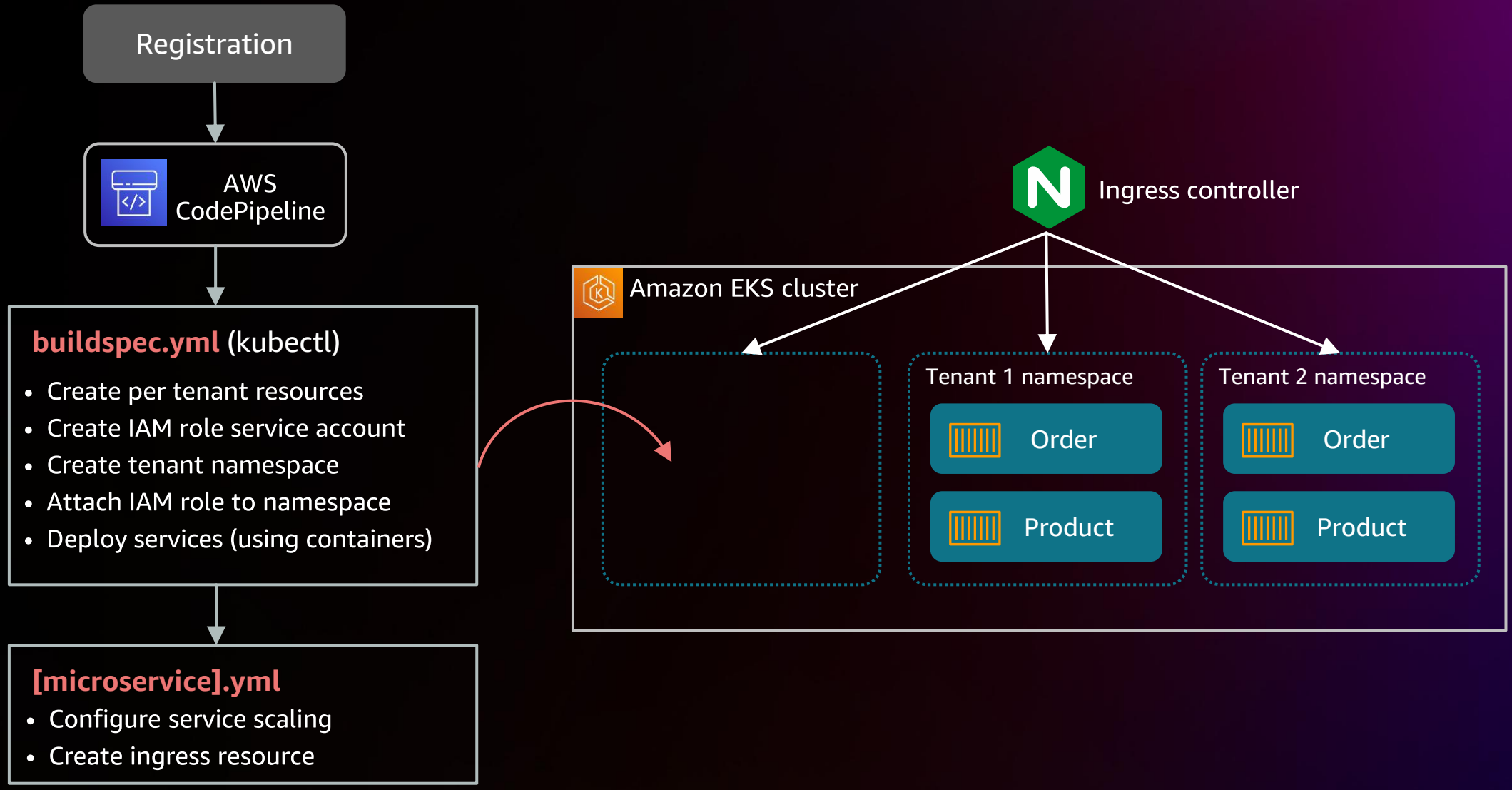
```
189     Type: AWS::CodeBuild::Project
190     Properties:
191       Artifacts:
192         Type: CODEPIPELINE
193       Source:
194         Type: CODEPIPELINE
195       Environment:
196         ComputeType: BUILD_GENERAL1_SMALL
197         Type: LINUX_CONTAINER
198         Image: !Ref CodeBuildDockerImage
199         EnvironmentVariables:
200           - Name: TENANT_NAME
201             Value: !Ref TenantName
202           - Name: CUSTOM_DOMAIN
203             Value: !Ref CustomDomain
204           - Name: EKS_CLUSTER_NAME
205             Value: !Ref EksClusterName
206           - Name: EKS_KUBECTL_ROLE_ARN
207             Value: !Sub arn:aws:iam::${AWS::AccountId}:role/${KubectlRoleName}
208           - Name: ACCOUNT_ID
209             Value: !Sub ${AWS::AccountId}
210           - Name: PRODUCT_SERVICE_ECR_REPO_URI
211             Value: !Ref ProductServiceEcrRepoUri
212           - Name: ORDER_SERVICE_ECR_REPO_URI
213             Value: !Ref OrderServiceEcrRepoUri
```

3 → buildspec.yml

Tenant deployment model



Namespace provisioning



Inside the tenant provisioning

buildspec.yml

1
```
kubectl create ns tenant1
```

2
```
aws dynamodb create-table --table-name Order-$TENANT_NAME --attribute-
definitions AttributeName=OrderId,AttributeType=S --key-schema
AttributeName=OrderId,KeyType=HASH --provisioned-throughput
ReadCapacityUnits=5,WriteCapacityUnits=5
```

3
```
kubectl apply -f services/application-services/productservice/kubernetes/
product-service.yaml -n tenant1
```

4
```
kubectl apply -f services/application-services/order-service/kubernetes/
order-service.yaml -n $TENANT_NAME
```

Microservice configuration/deployment

Order-service.yaml

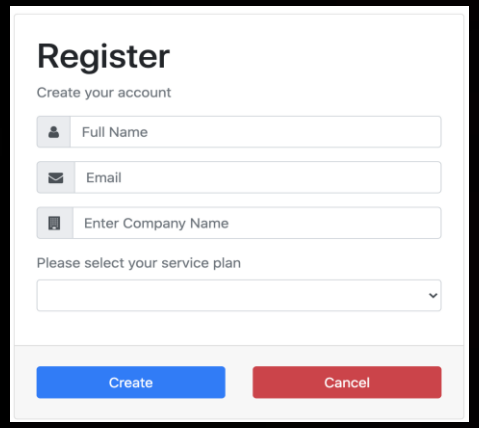
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: order
spec:
  replicas: 1
  selector:
    matchLabels:
      app: order
  template:
    metadata:
      labels:
        app: order
    spec:
      containers:
      - name: order
        image: ORDER_SERVICE_ECR_REPO_URI:latest
        ports:
        - containerPort: 5001
          name: "http"
```

Deployment

```yaml
apiVersion: v1
kind: Service
metadata:
  name: order-service
spec:
  selector:
    app: order
  ports:
  - name: http
    protocol: TCP
    port: 80
    targetPort: 5001
  type: NodePort
```

Service

Provisioning tenant subdomain




```
{
  "fullName": "John Doe",
  "companyName": "Tenant1",
  "email": "jdoe@saasco.com",
  "plan": "basic"
}
```

Amazon CloudFront

- Retrieve distro
- Add alias

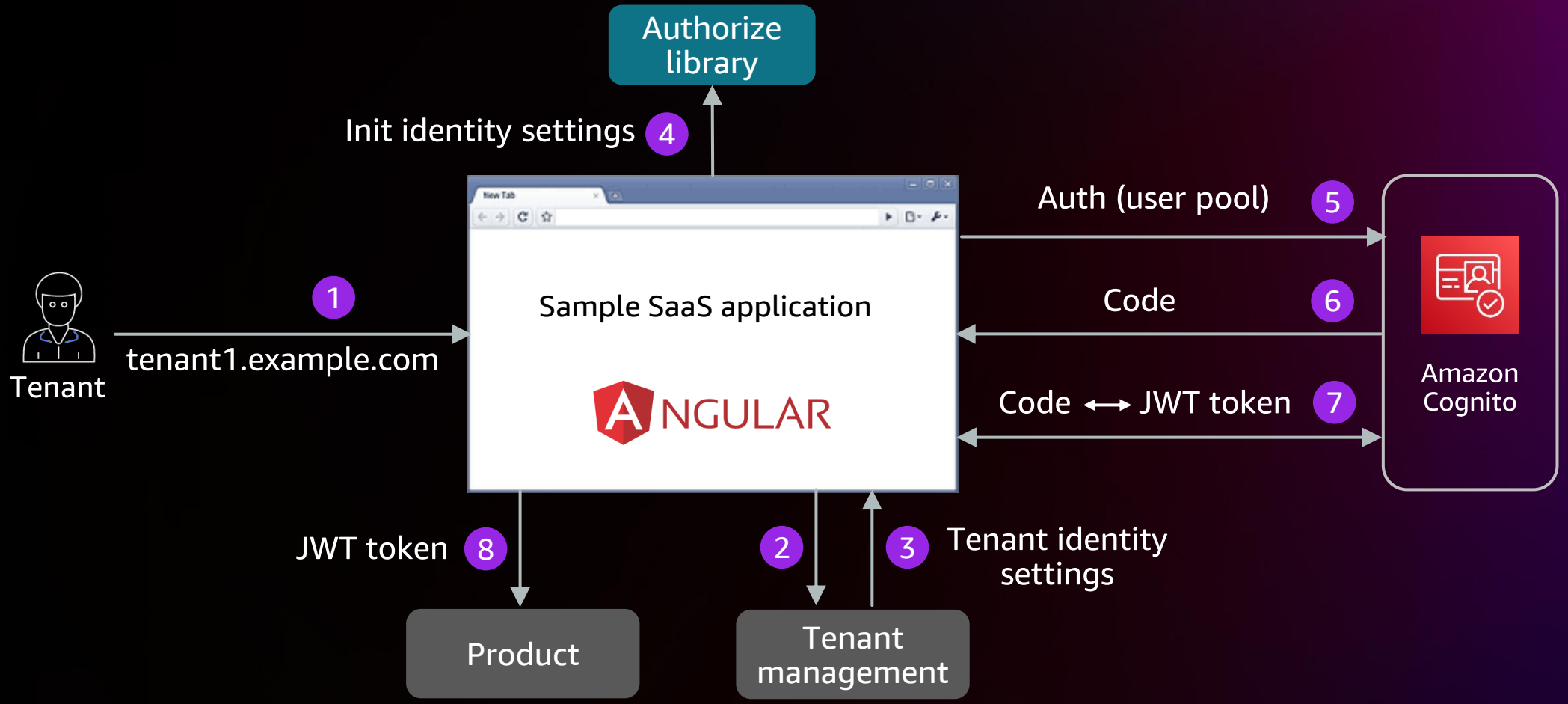
Origin	DomainName	AlternateNames
App-Bucket	https://abc123.cloudfront.net	app.saasco.com
		tenant1.saasco.com

Amazon Route 53

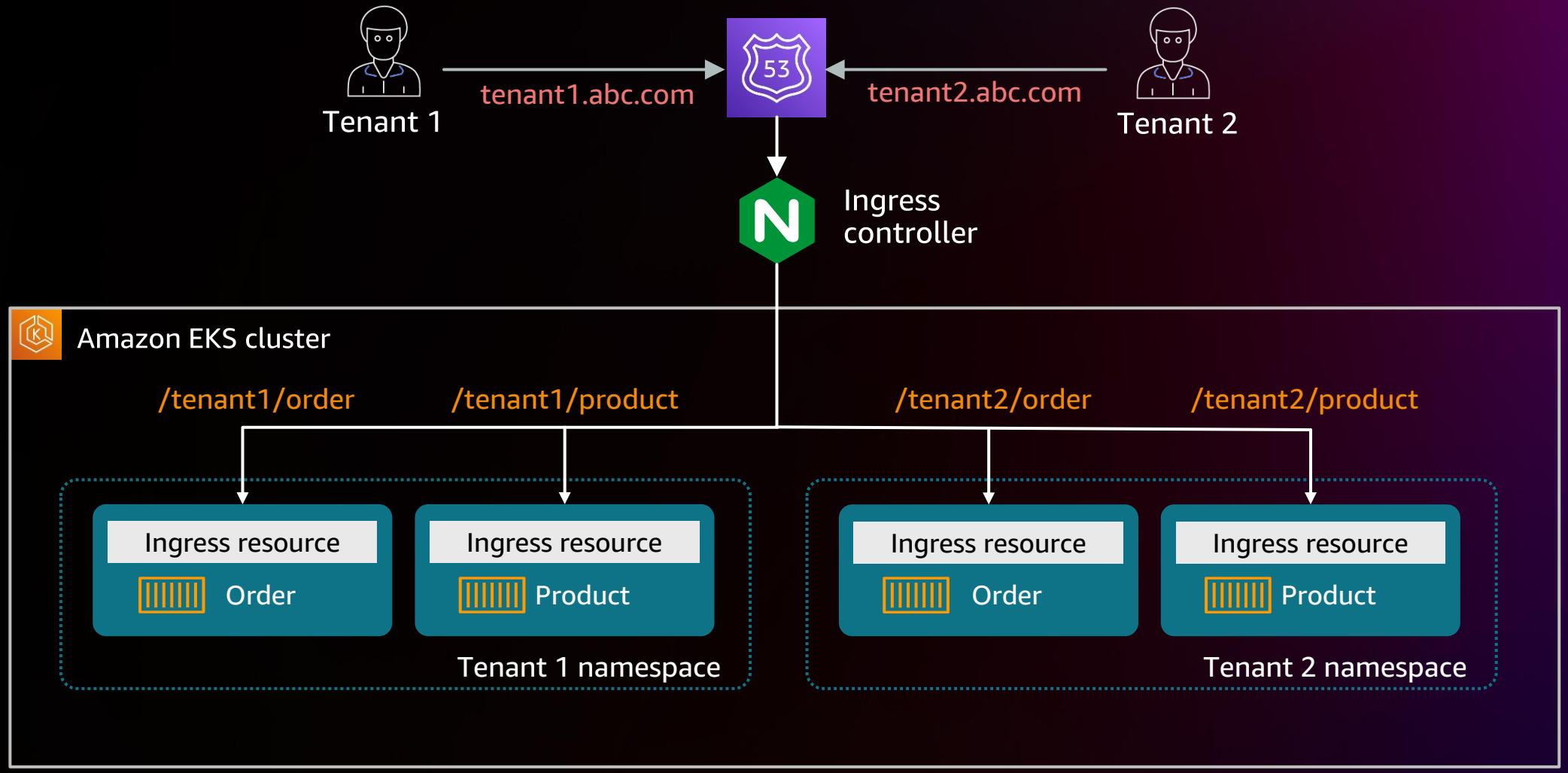
- Create RecordSet

Record Name	Type	Route to
app.saasco.com	A	https://abc123.cloudfront.net
tenant1.saasco.com	A	https://abc123.cloudfront.net

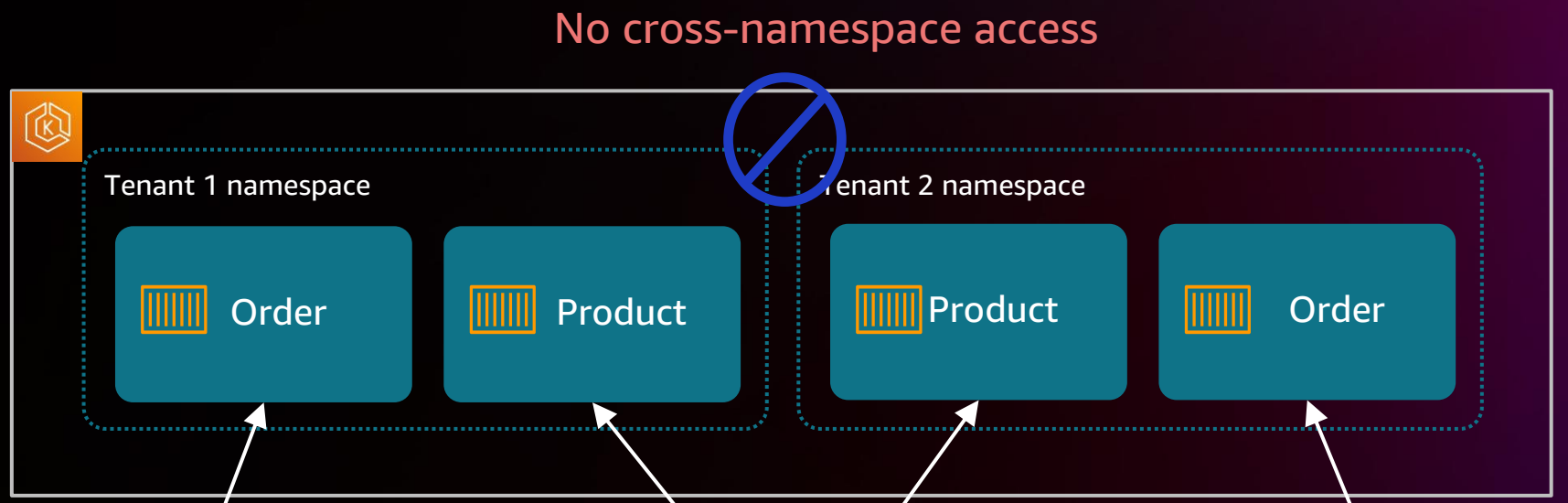
Tenant authentication



Routing tenants to namespaces



Tenant isolation



No cross-tenant access to resources

Product ID	Price	Quantity
33919	$50.00	10
98194	$29.99	34
73919	$104.12	2

Order-Tenant1 table

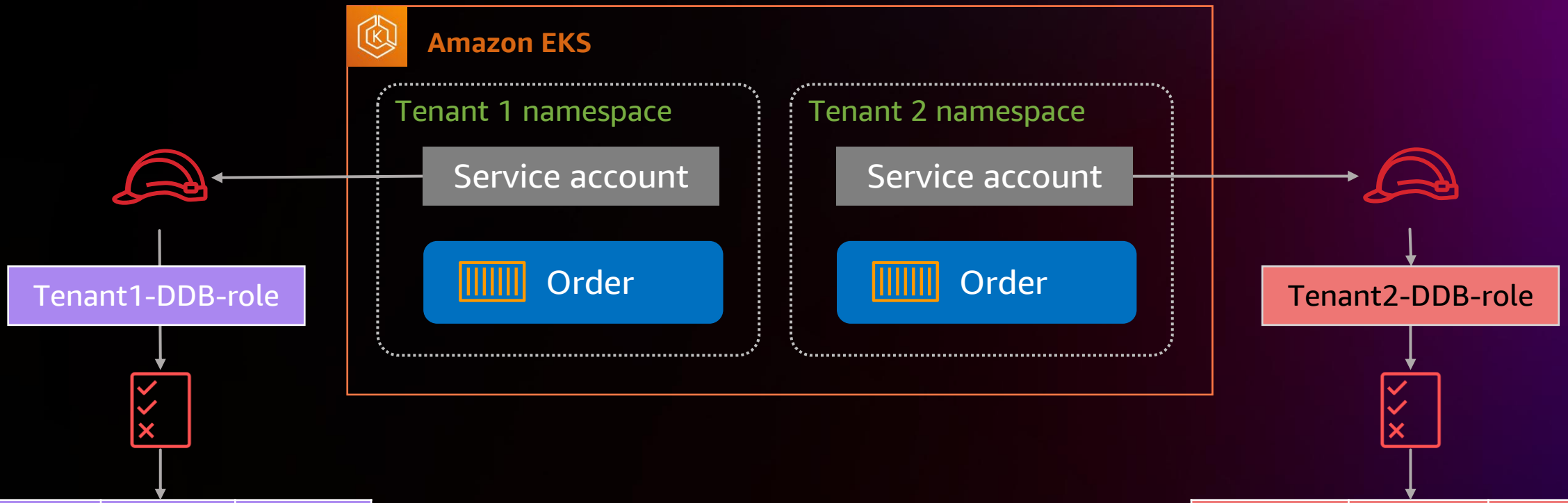
Partition Key	SKU	Price
Tenant1	848181	$12.48
Tenant4	761991	$9.14
Tenant1	484188	$141.81

Product table

Product ID	Price	Quantity
33919	$50.00	10
98194	$29.99	34
73919	$104.12	2

Order-Tenant2 table

Applying IAM roles for service accounts (IRSA)



OrderId	Price	Quantity
20001	$500.00	20
20002	$299.99	24
20003	$204.12	30

Order-Tenant1 table

OrderId	Price	Quantity
33919	$13.55	1
98194	$11.66	34
73919	$532.33	2

Order-Tenant2 table

IAM policy for order tables

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "TENANT_NAME",
            "Effect": "Allow",
            "Action": "dynamodb:*",
            "Resource": "arn:aws:dynamodb:us-east-1:ACCOUNT_ID:table/Order-TENANT_NAME"
        }
    ]
}
```

Scope access to order table by Tenant ID

Order service: GetOrderById()

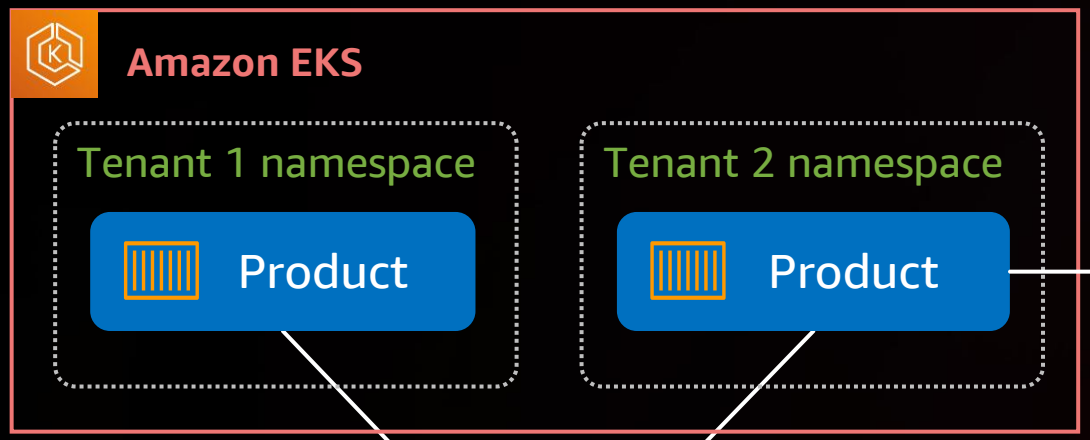
```
84  @GetMapping(value = "{companyName}/order/api/order/{orderId}", produces = { MediaType.APPLICATION_JSON_VALUE })
85  public Order getOrderById(@PathVariable("orderId") String orderId, HttpServletRequest request) {
86    String tenantId = null;
87    Order order = null;
88
89    try {
90      tenantId = tokenManager.getTenantId(request);
91
92      if (tenantId != null && !tenantId.isEmpty()) {
93        order = orderService.getOrderById(orderId, tenantId);
94        return order;
95      }
96    } catch (Exception e) {
97      logger.error("TenantId: " + tenantId + "-get order by ID failed: ", e);
98      return null;
99    }
100
101   return order;
102 }
```

Map tenant to tenant siloed table

```
113  public DynamoDBMapper dynamoDBMapper(String tenantId) {
114      String tableName = "Order-" + tenantId;
115      DynamoDBMapperConfig dbMapperConfig = new DynamoDBMapperConfig.Builder()
116          .withTableNameOverride(TableNameOverride.withTableNameReplacement(tableName)).build();
117
118      AmazonDynamoDBClient dynamoClient = getAmazonDynamoDBLocalClient(tenantId);
119      return new DynamoDBMapper(dynamoClient, dbMapperConfig);
120  }
```

```
83      DynamoDBMapperConfig config = DynamoDBMapperConfig.builder()
84          .withConsistentReads(DynamoDBMapperConfig.ConsistentReads.CONSISTENT).build();
85      try {
86        order = mapper.load(Order.class, orderId, config);
87      } catch (Exception e) {
88        logger.error("TenantId: " + tenantId + "-Get Order By Id failed " + e.getMessage());
89      }
```

Pool isolation for products

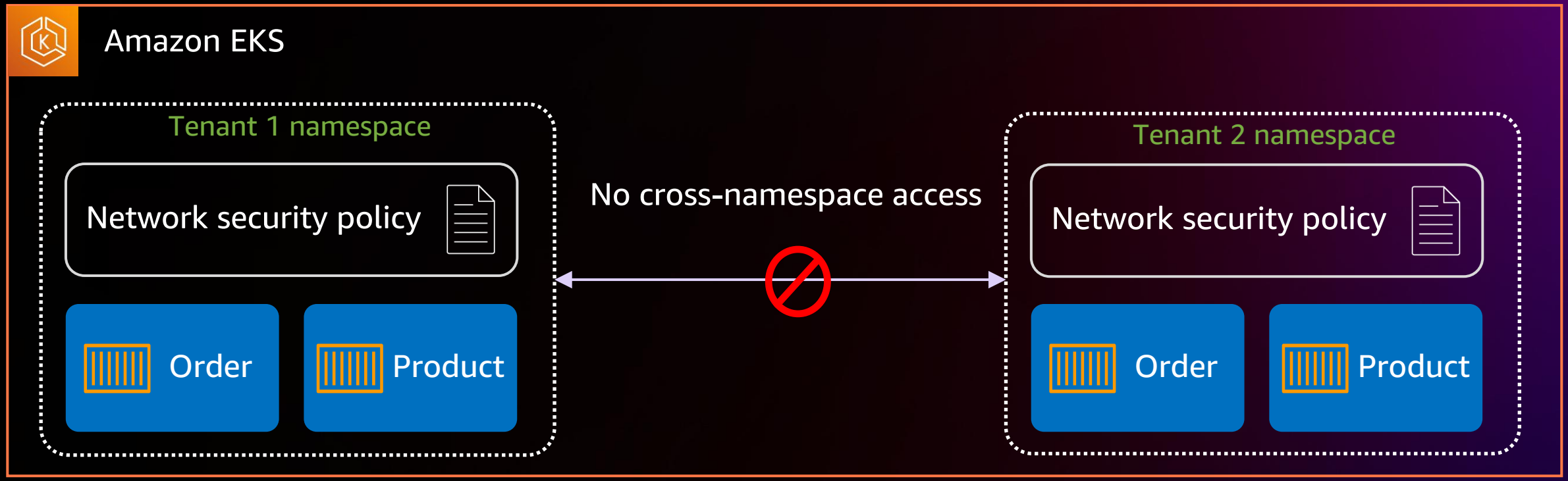


PartitionKey	SKU	Price
Tenant1	848181	$12.48
Tenant4	761991	$9.14
Tenant1	484188	$141.81

Product table

```
{
  "Sid": "TenantReadOnlyOrderTable",
  "Effect": "Allow",
  "Action": [
      "dynamodb:GetItem",
      "dynamodb:BatchGetItem",
      "dynamodb:Query",
      "dynamodb:DescribeTable"
  ],
  "Resource": [
      "arn:aws:dynamodb:[region]:table/Product"
  ],
  "Condition": {
      "ForAllValues:StringEquals": {
          "dynamodb:LeadingKeys": [
              "tenant1"
          ]
      }
  }
}
```

Isolating namespaces



Isolation policies

```yaml
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: TENANT_NAME
  name: TENANT_NAME-policy-deny-other-namespace
spec:
  podSelector:
    matchLabels:
  ingress:
  - from:
    - podSelector: {}
```

Network policy prevents cross-namespace access

Isolation policy configuration

buildspec.yml

1
```
aws iam create-policy --policy-name $TENANT_NAME-ddb-policy --policy-
document file://resources/policy/order-ddb-table-policy.json
```

2
```
eksctl create iamserviceaccount --name $TENANT_NAME-ddb-sa --
namespace=$TENANT_NAME --cluster $EKS_CLUSTER_NAME --attach-policy-arn
arn:aws:iam::$ACCOUNT_ID:policy/$TENANT_NAME-ddb-policy --approve
```

3
```
kubectl apply -f resources/policy/tenant-service-policy.yaml -n tenant1
```

Takeaways

- Amazon EKS aligns well the demands and value prop of SaaS environments
- Consider your deployment model(s) carefully
- Expect to support a mix of deployment models
- Find the routing strategy that best aligns to your needs
- Take advantage of fine-graining scaling/tiering constructs
- Isolate at the network and resource level
- Use shared libraries to hide multi-tenant mechanisms
- Lean on the Kubernetes community and partner solutions

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
- SAS302 – Supporting extensibility in SaaS Environments
- 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 – 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

Builders' 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!

Tod Golding
todg@amazon.com

Toby Buckley
tobuck@amazon.com

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