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

SAS306

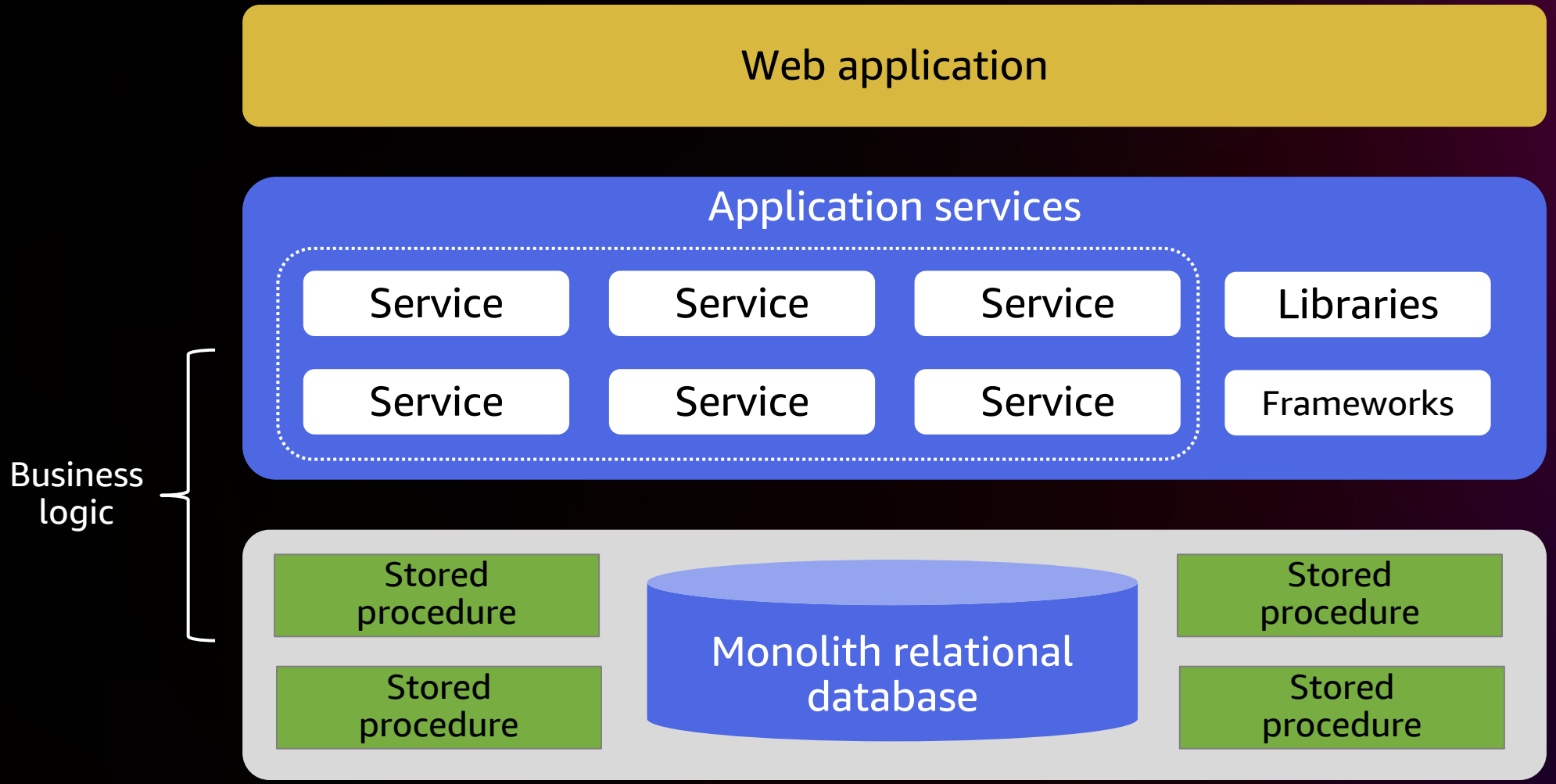
SaaS migration: Inside a real-world multi-tenant transformation

Tod Golding

Senior Principal Solutions Architect,
AWS SaaS Factory
AWS

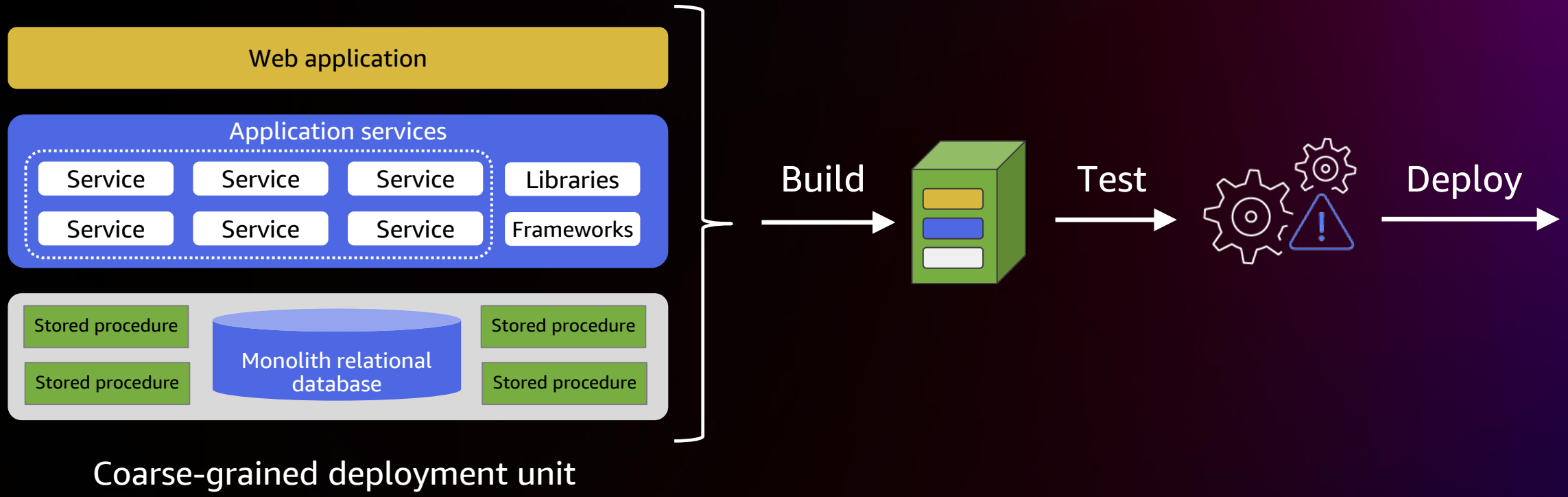
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Where we're starting



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

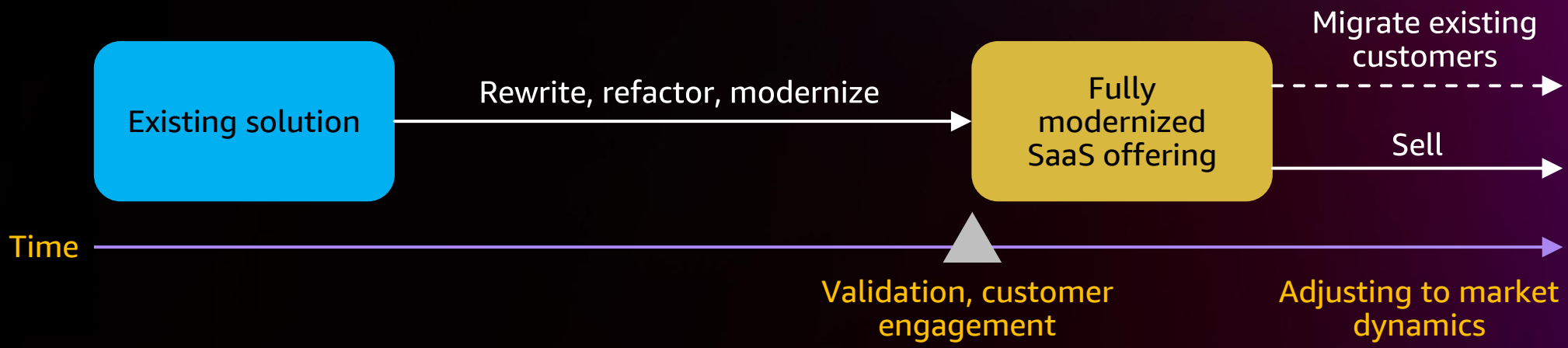
A highly coupled build and deployment



Coarse-grained deployment unit

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

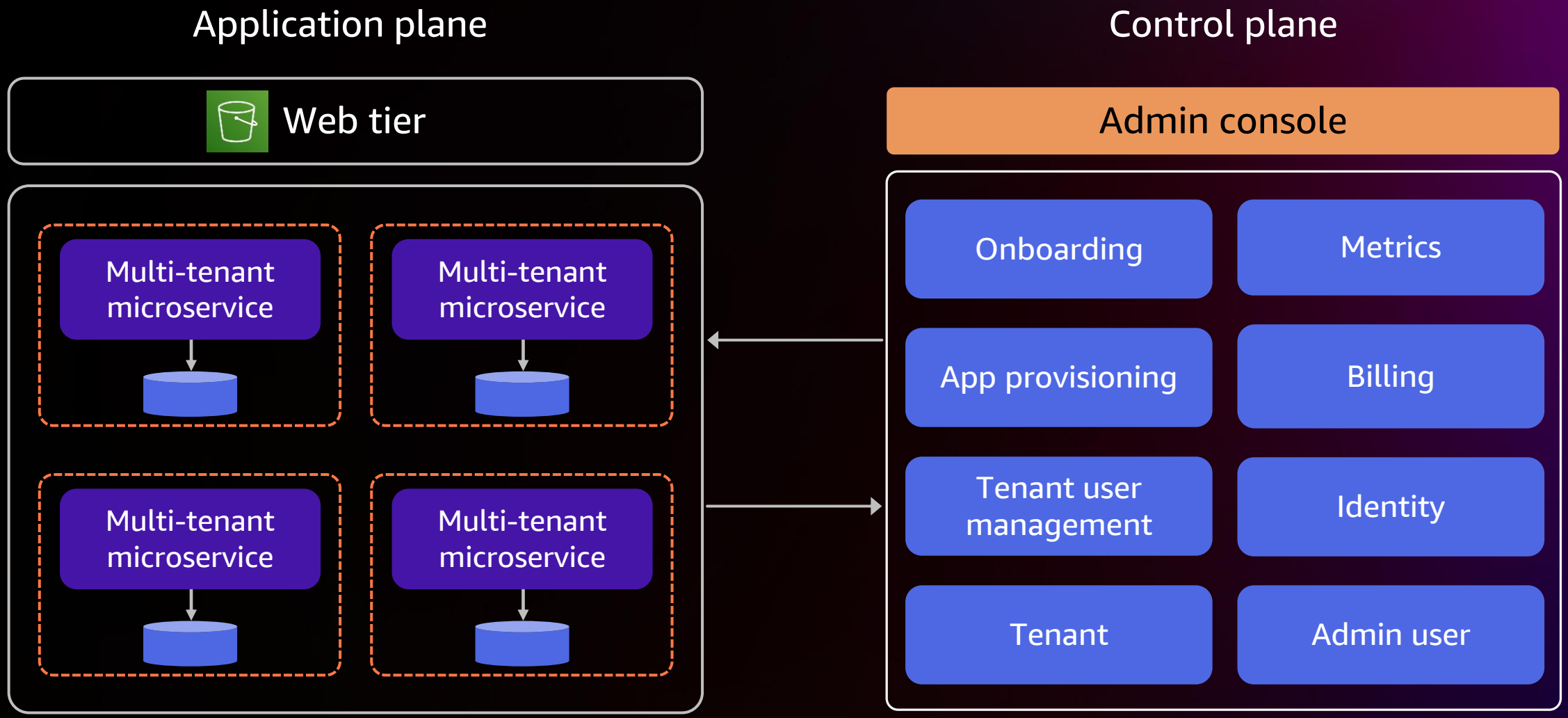
Our migration strategy





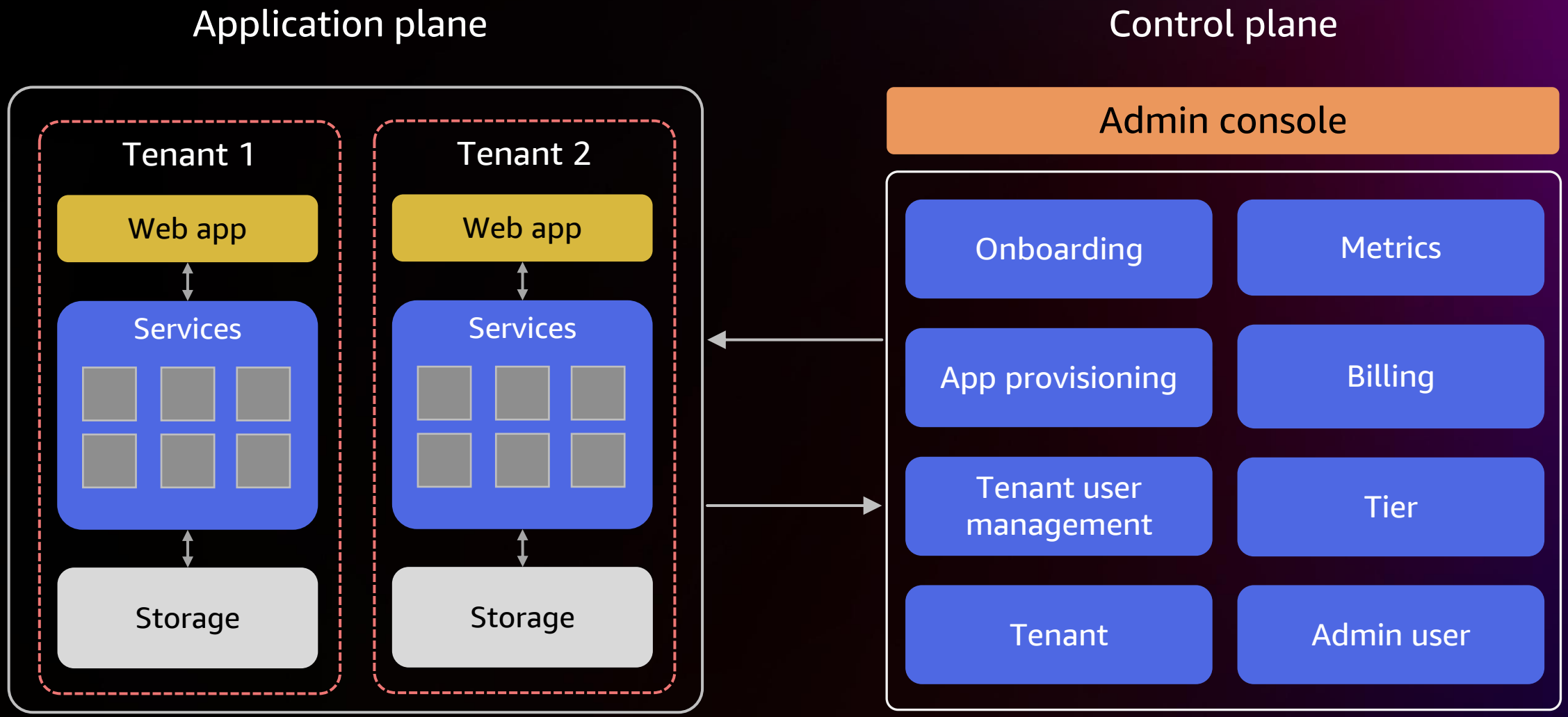
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Our target end state



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Adopting a full stack silo model

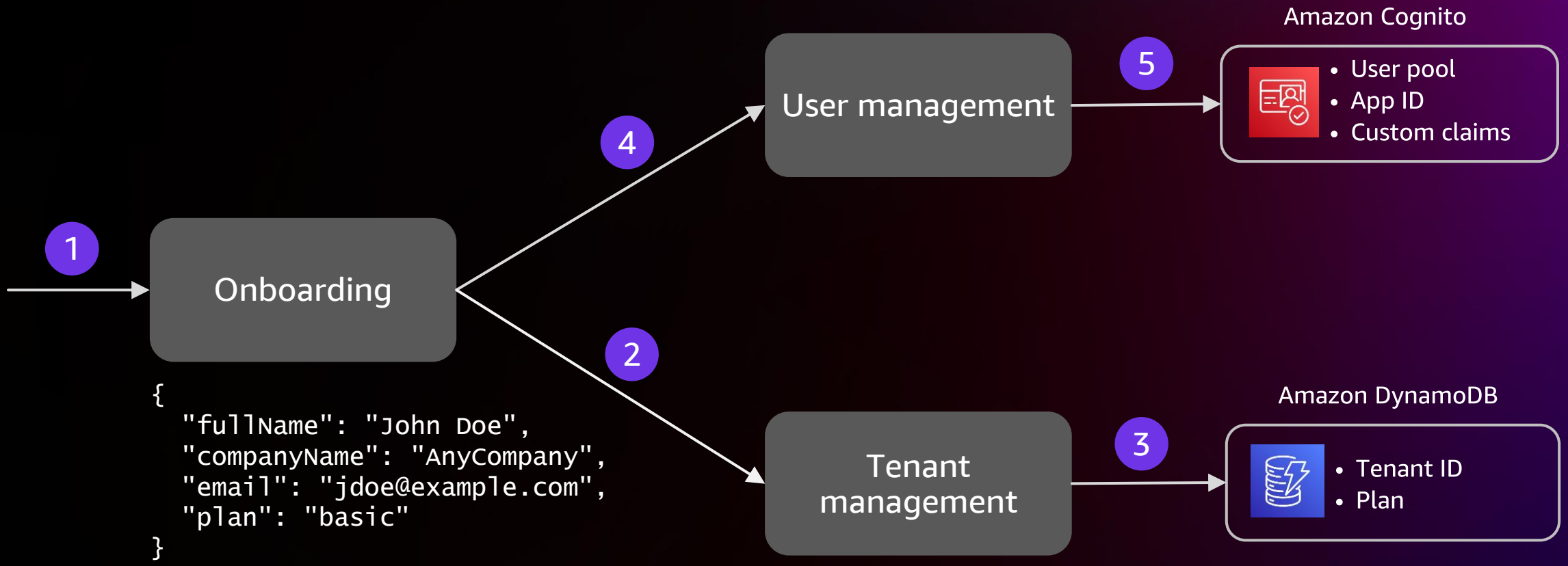


© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Bringing your control plane to life

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Let’s build a lightweight control plane



Enabling the introduction of tenants and tenant identity

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Inside tenant registration

```
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    }
```

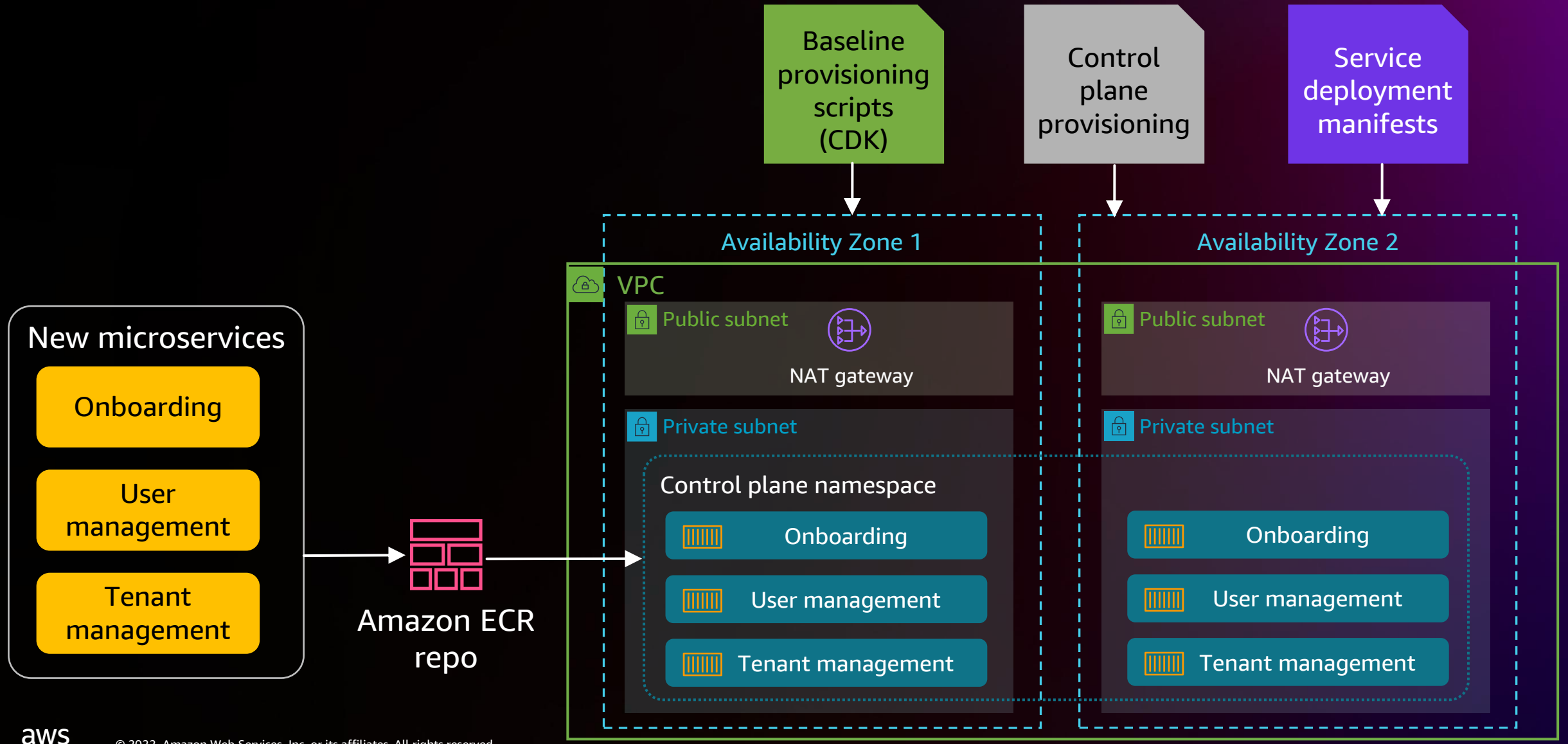
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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  }
```

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Provisioning the baseline environment



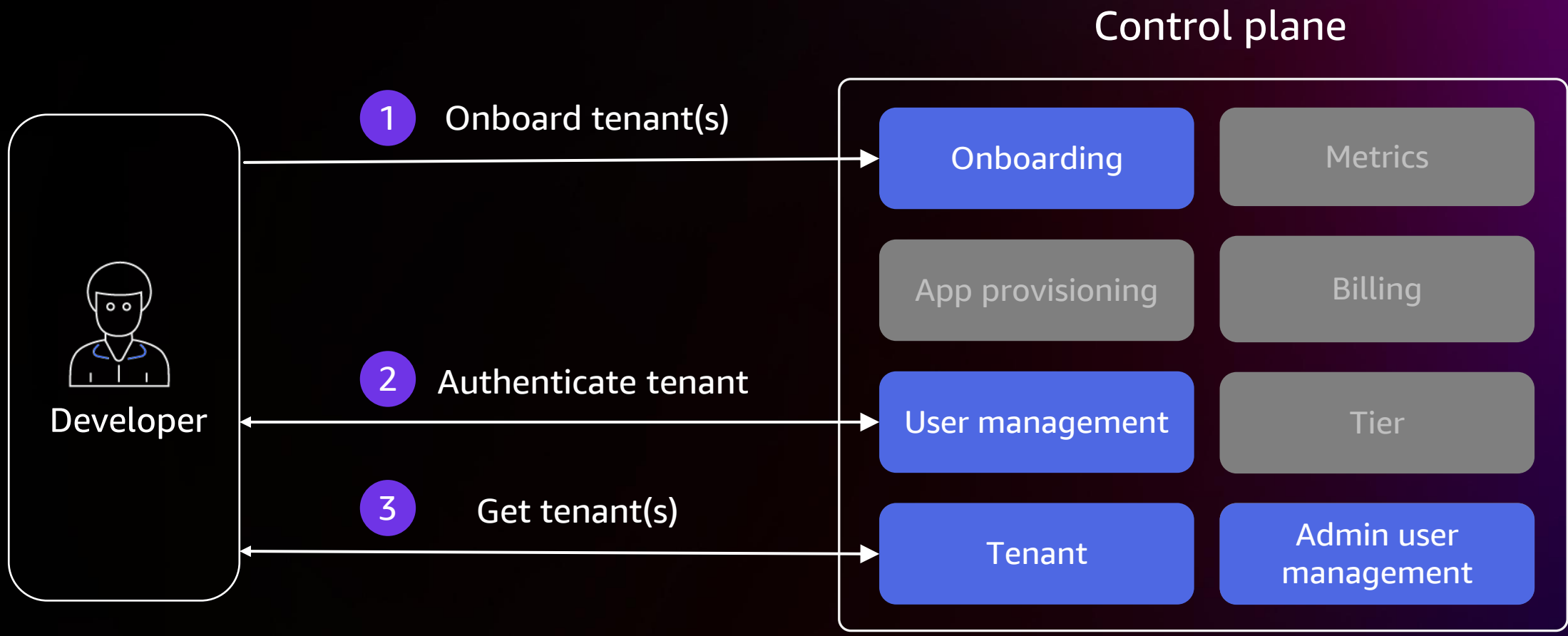
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Creating a system admin user



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Exercise tenant onboarding

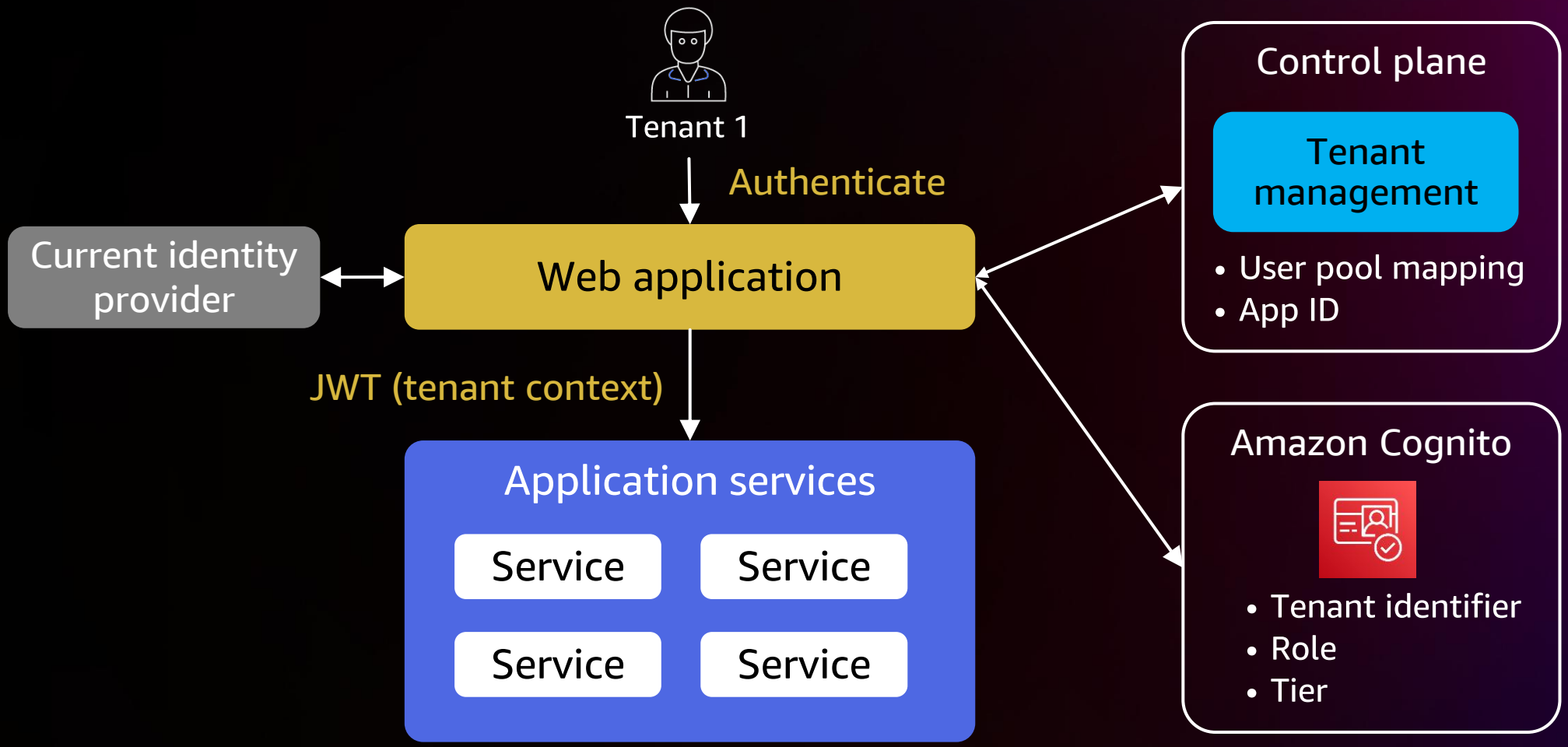


aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Adding and enabling tenant context

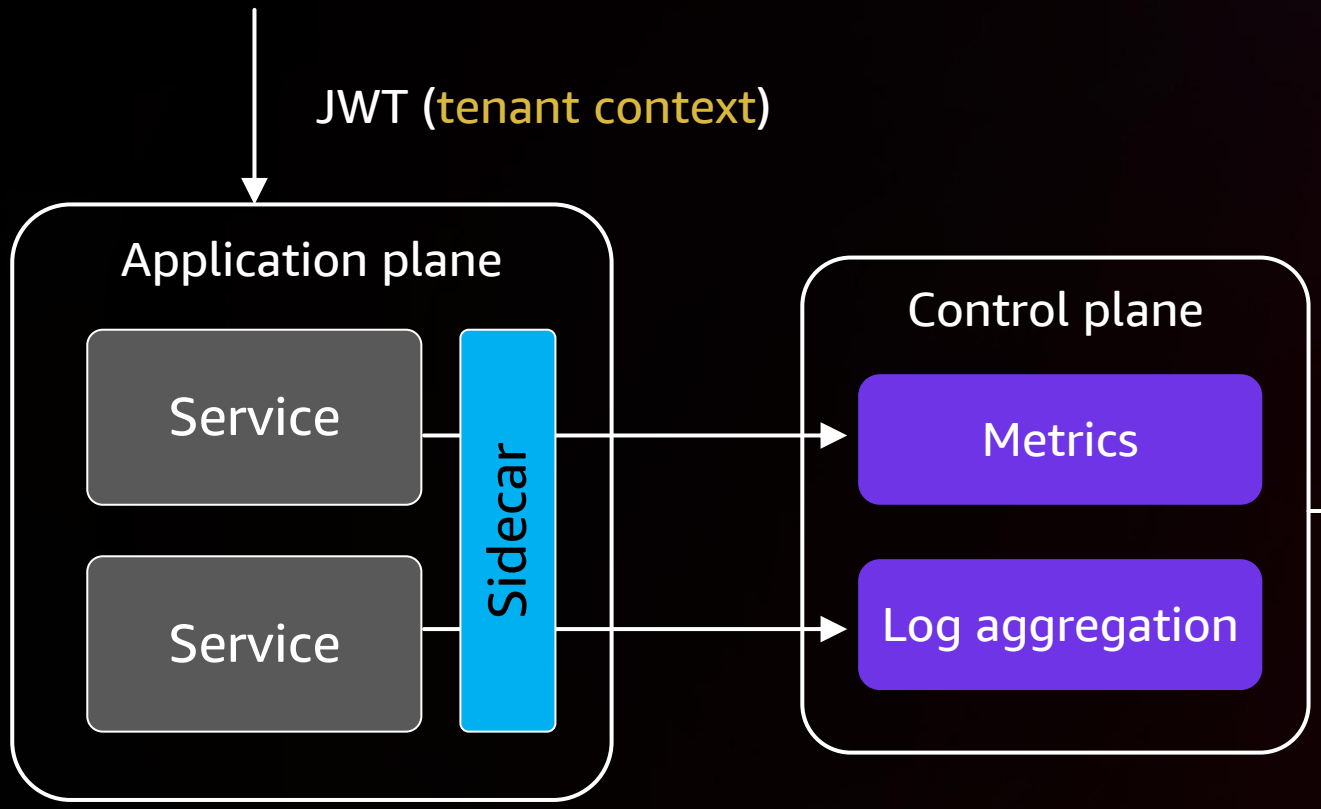
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Porting identity and injecting tenant context



aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Enabling multi-tenant operations

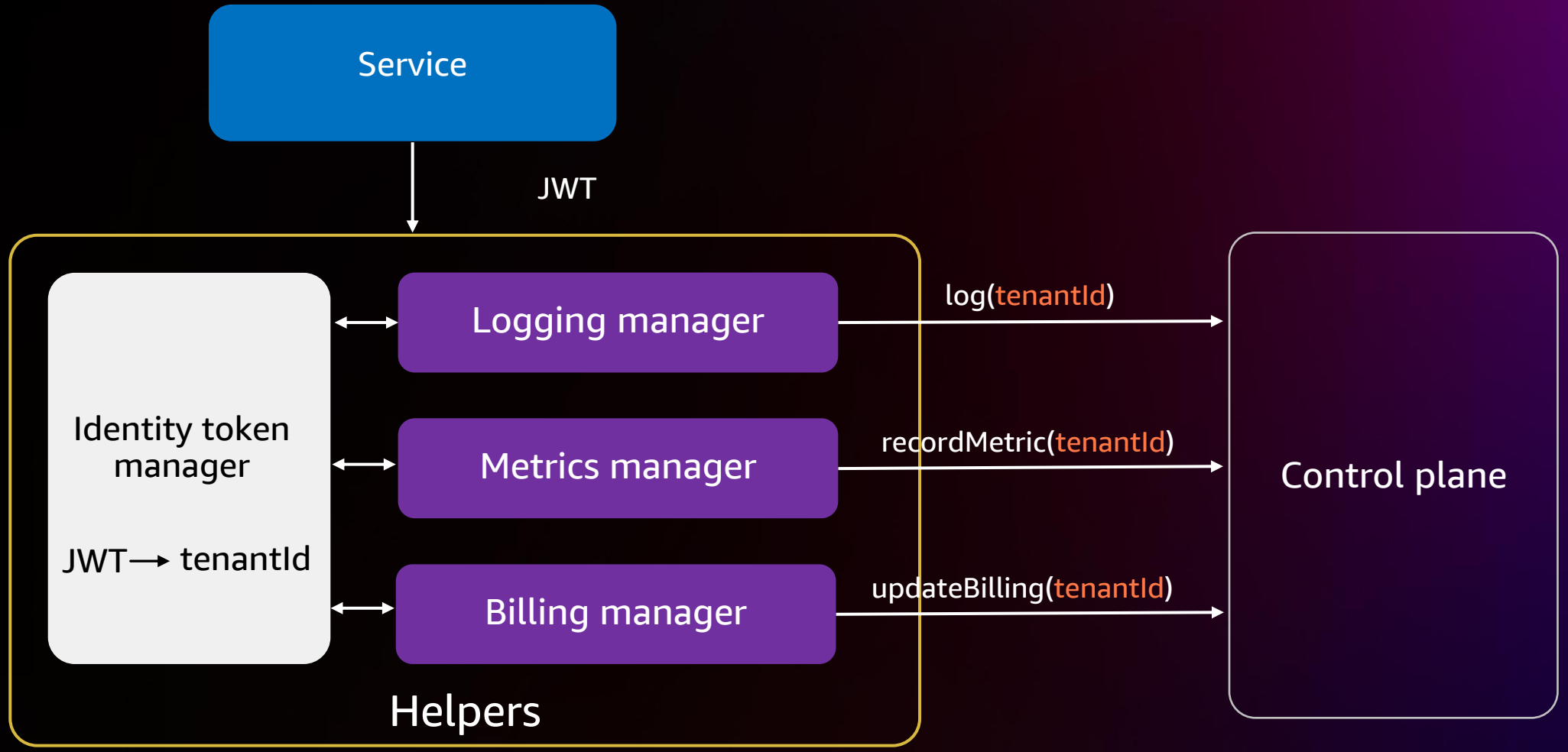




Tenant-aware operations

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Using helpers to make this less invasive



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

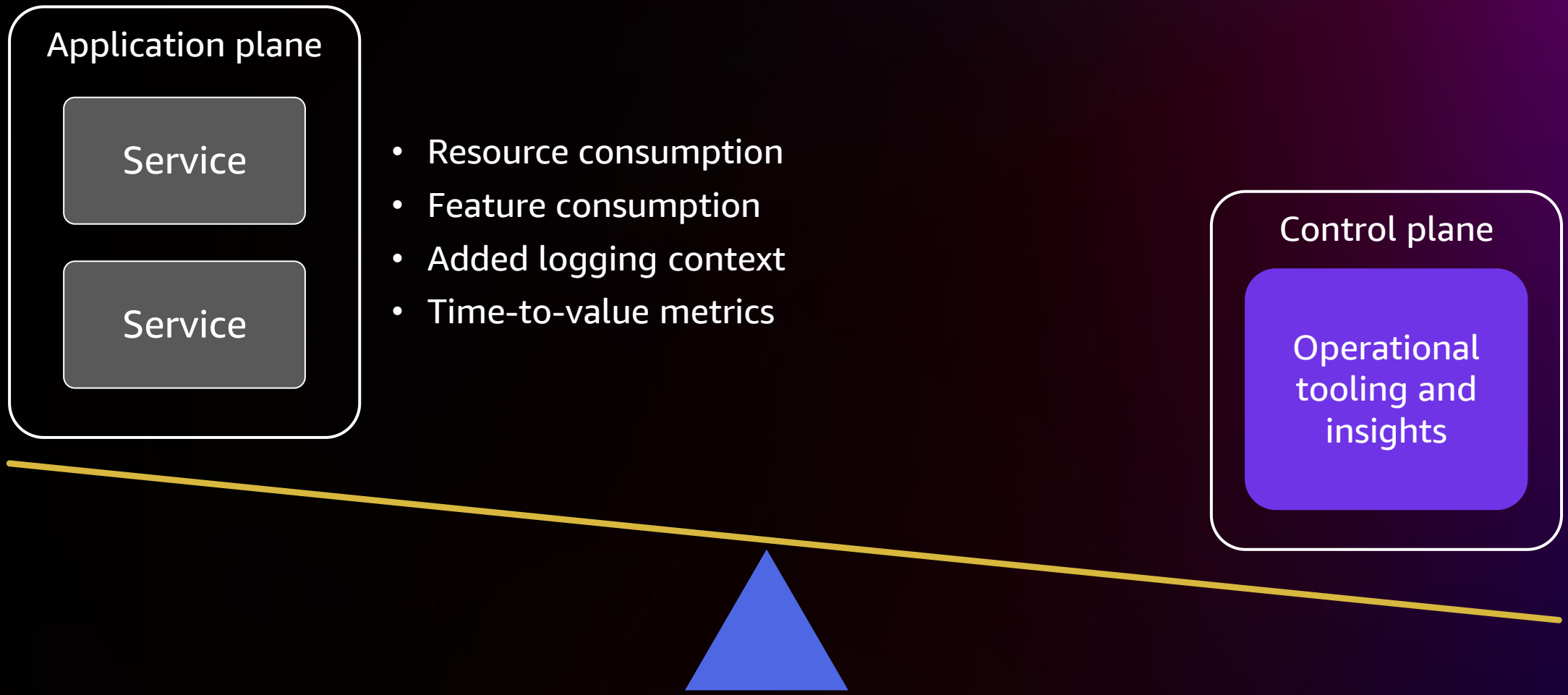
Accessing tenant context

```
117   @PostMapping(value = "{companyName}/order/api/order", produces = { MediaType.APPLICATION_JSON_VALUE })
118   public Order saveOrder(@RequestBody Order order, HttpServletRequest request) {
119     String tenantId = null;
120     Order newOrder = null;
121
122     try {
123       tenantId = tokenManager.getTenantId(request);
124       if (tenantId != null && !tenantId.isEmpty()) {
125         newOrder = orderService.save(order, tenantId);
126         return newOrder;
127       }
128     } catch (Exception e) {
129           LogManager.logMessage("save order failed: ");
130       return null;
131     }
132
133     return newOrder;
134   }
```

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How much to invest in legacy services?

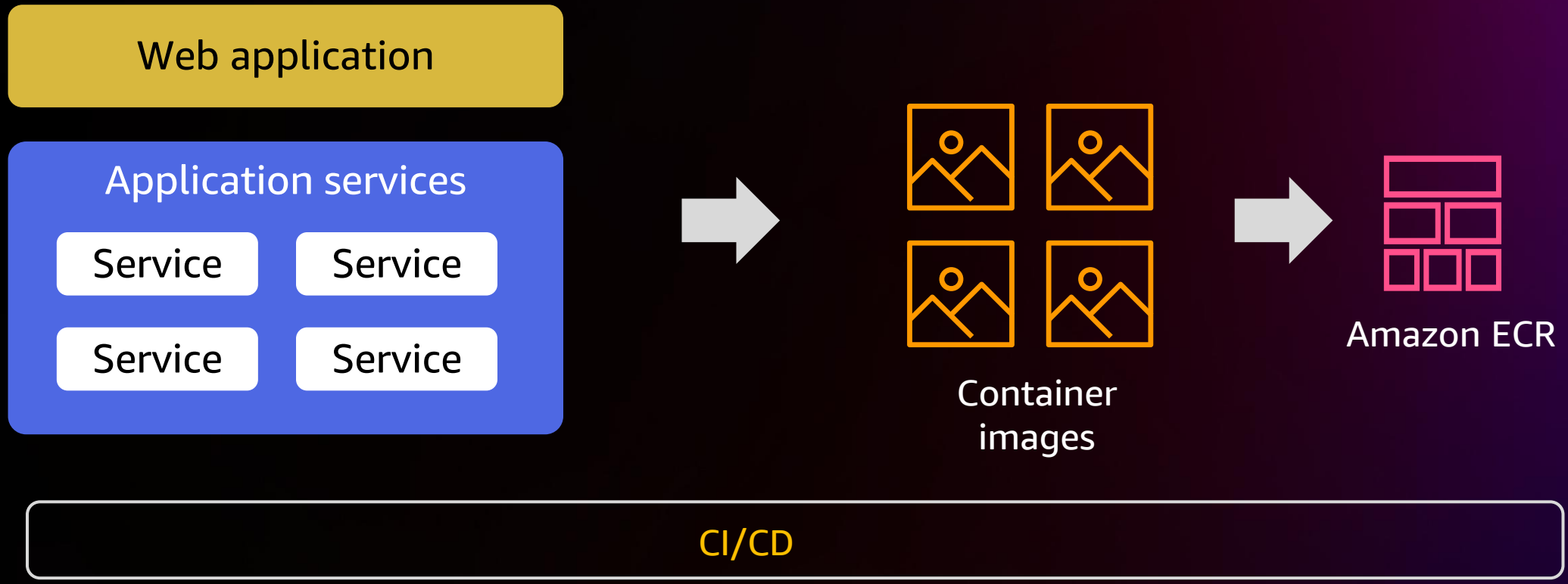


© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Provisioning your tenant siloes

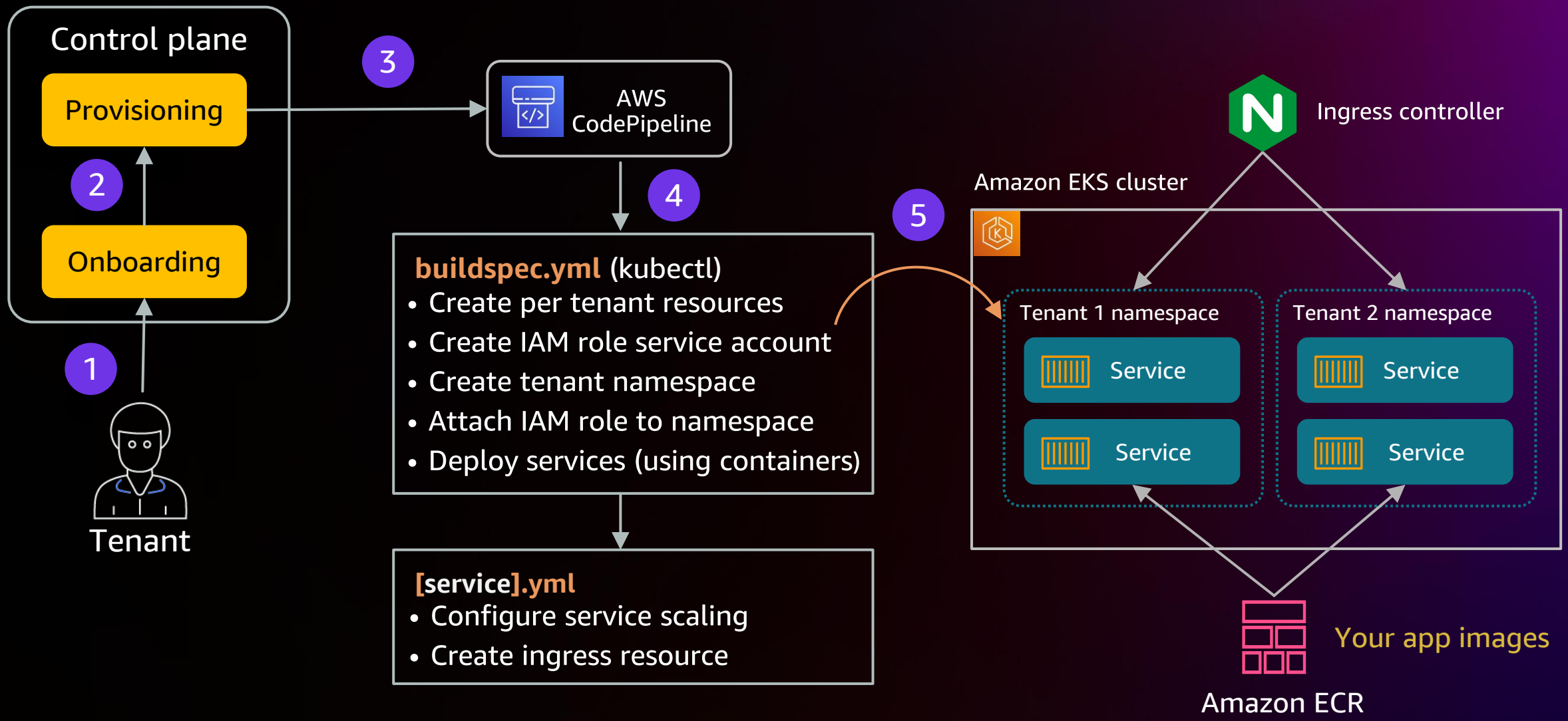
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Containerizing your application



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Provisioning tenant siloes



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Automating application provisioning

buildspec.yml

1. `kubectl create ns tenant1`

2. `kubectl apply -f services/application-services/service1/kubernetes/service1.yaml -n tenant1`

3. `kubectl apply -f services/application-services/service2/kubernetes/service2.yaml -n tenant2`

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Microservice configuration/deployment

your-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

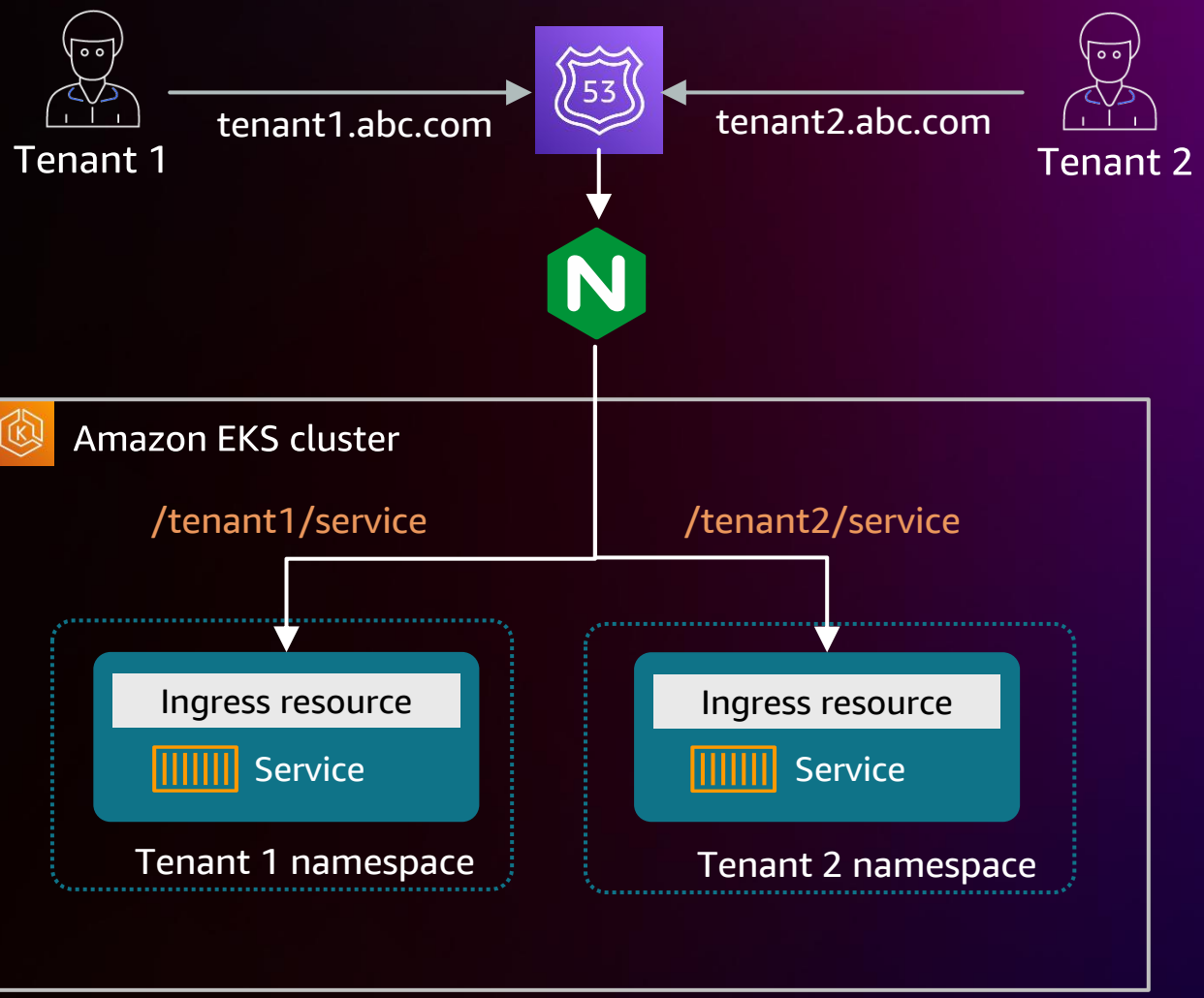
aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Routing tenants to each namespace

your-service.yaml

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: order-service-ingress
  annotations:
    kubernetes.io/ingress.class: "nginx"
spec:
  rules:
    - host: api.CUSTOM_DOMAIN
      http:
        paths:
          - path: /TENANT_NAME/order
            backend:
              service:
                name: order-service
                port:
                  number: 80
            pathType: Prefix
```

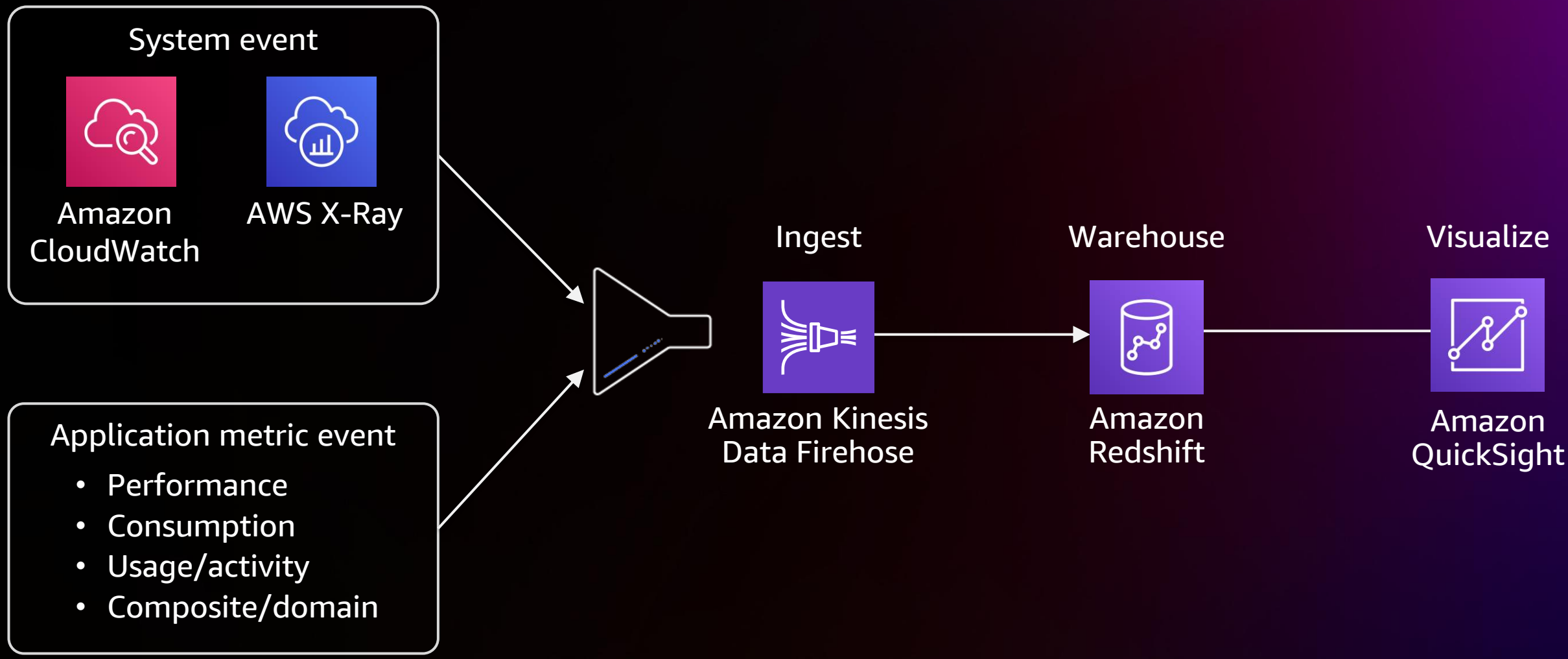


aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Enriching your control plane

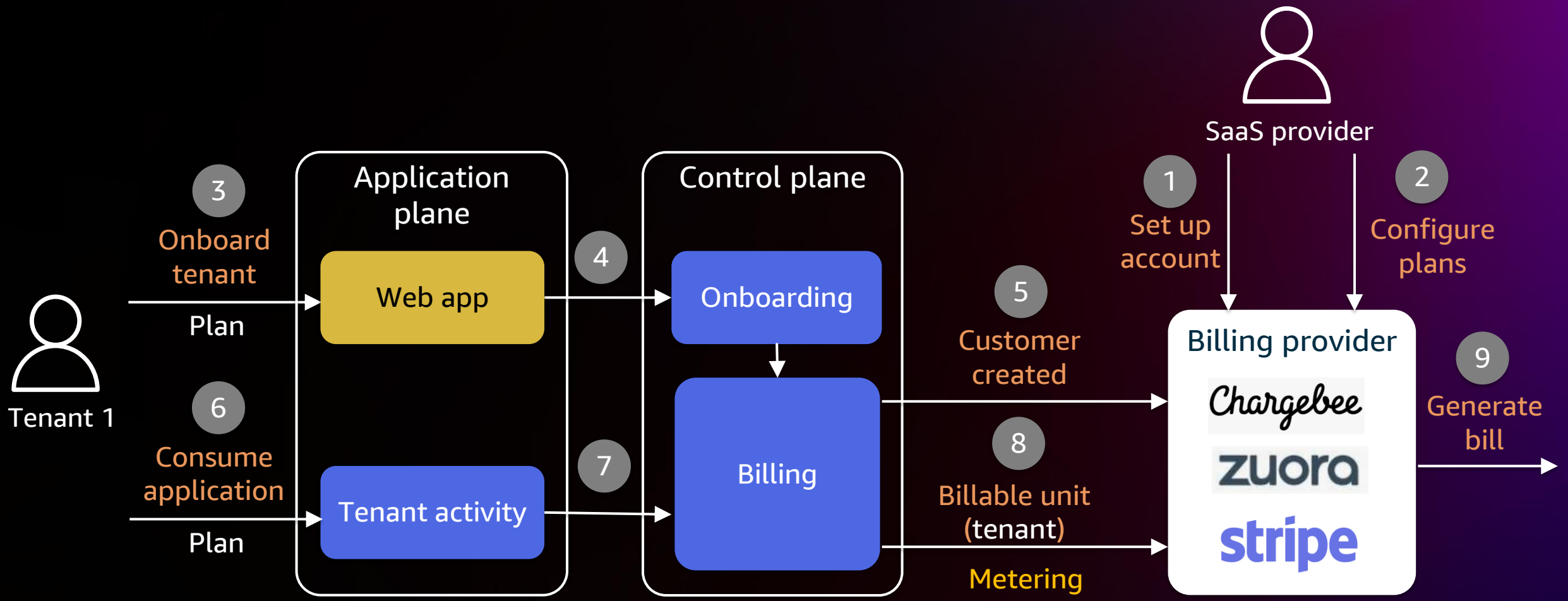
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Metrics aggregation



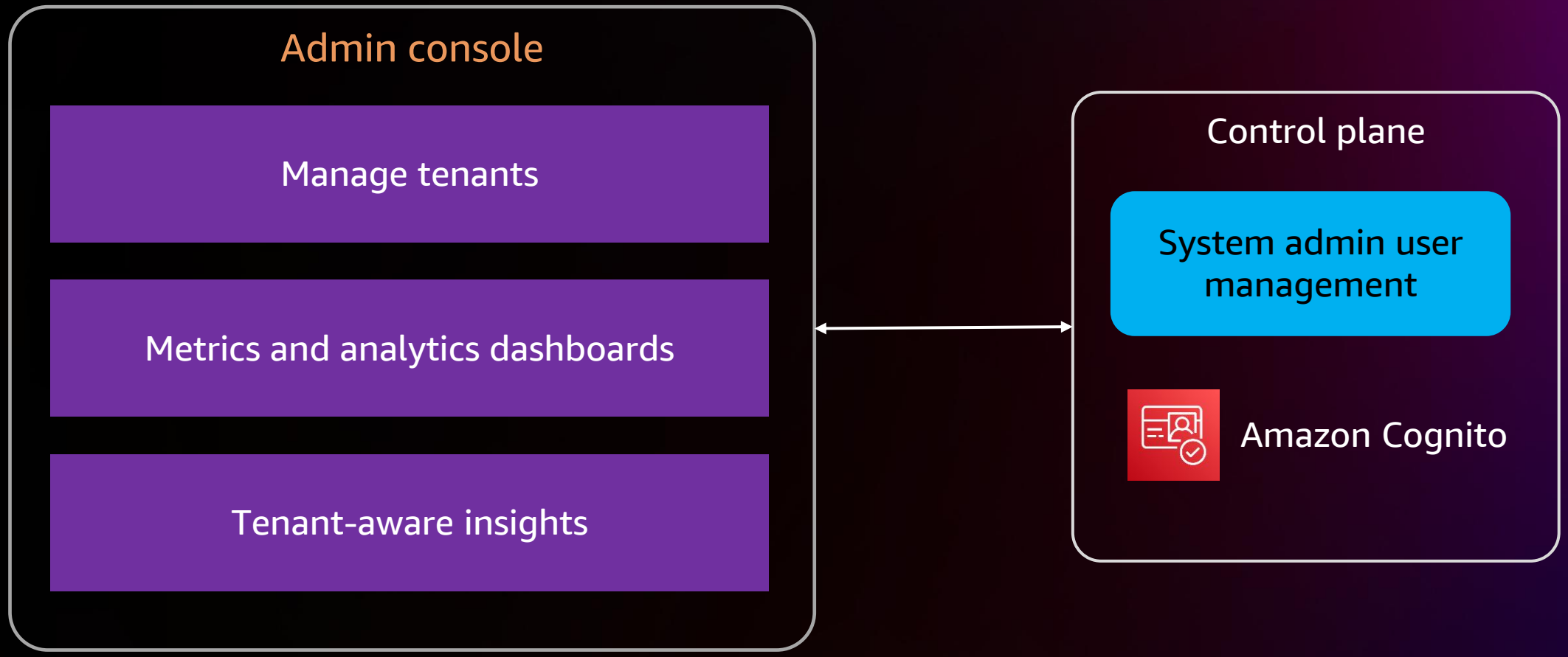
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Supporting billing integration



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Creating a system admin console



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

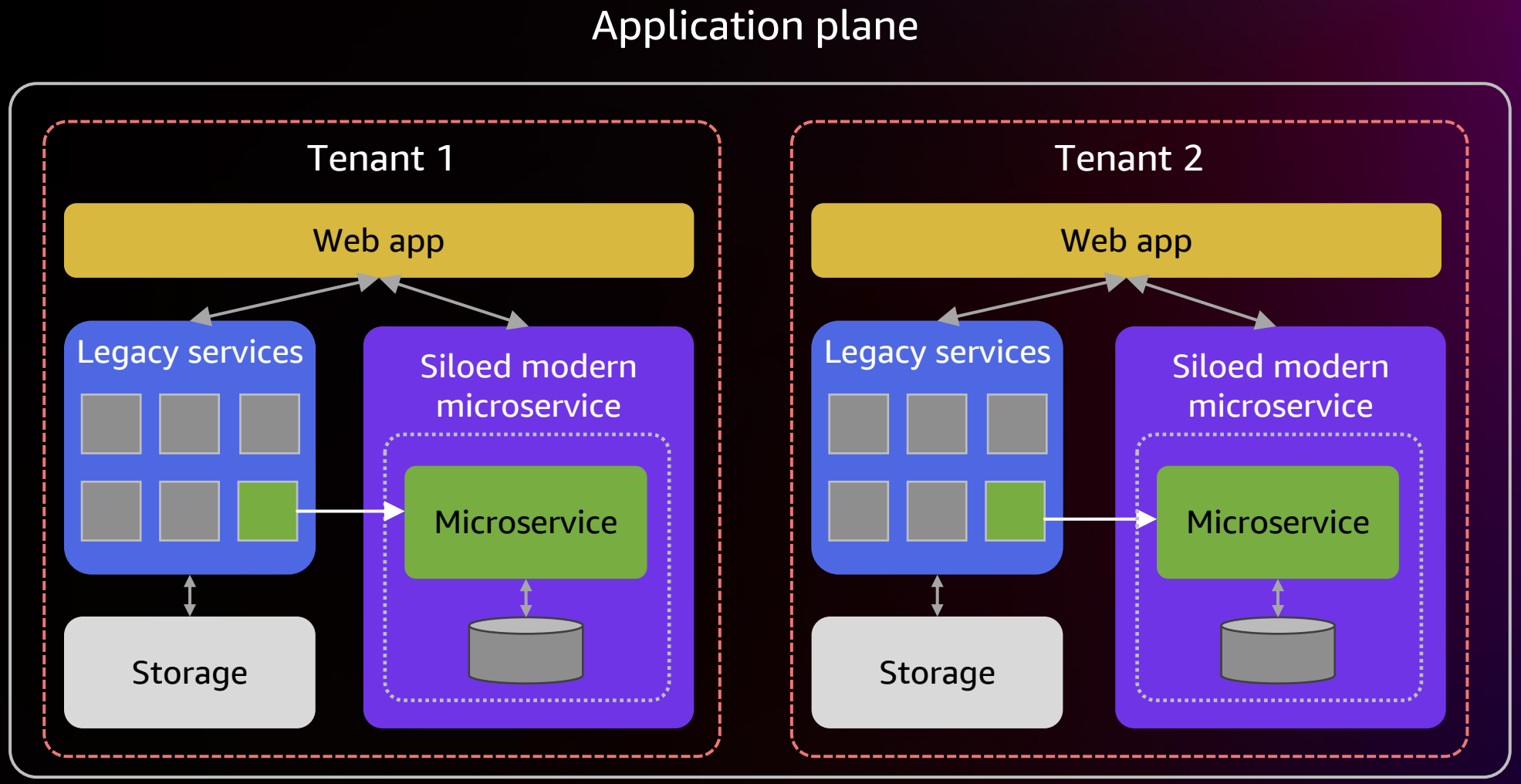
Your MVP is ready

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Modernizing your environment

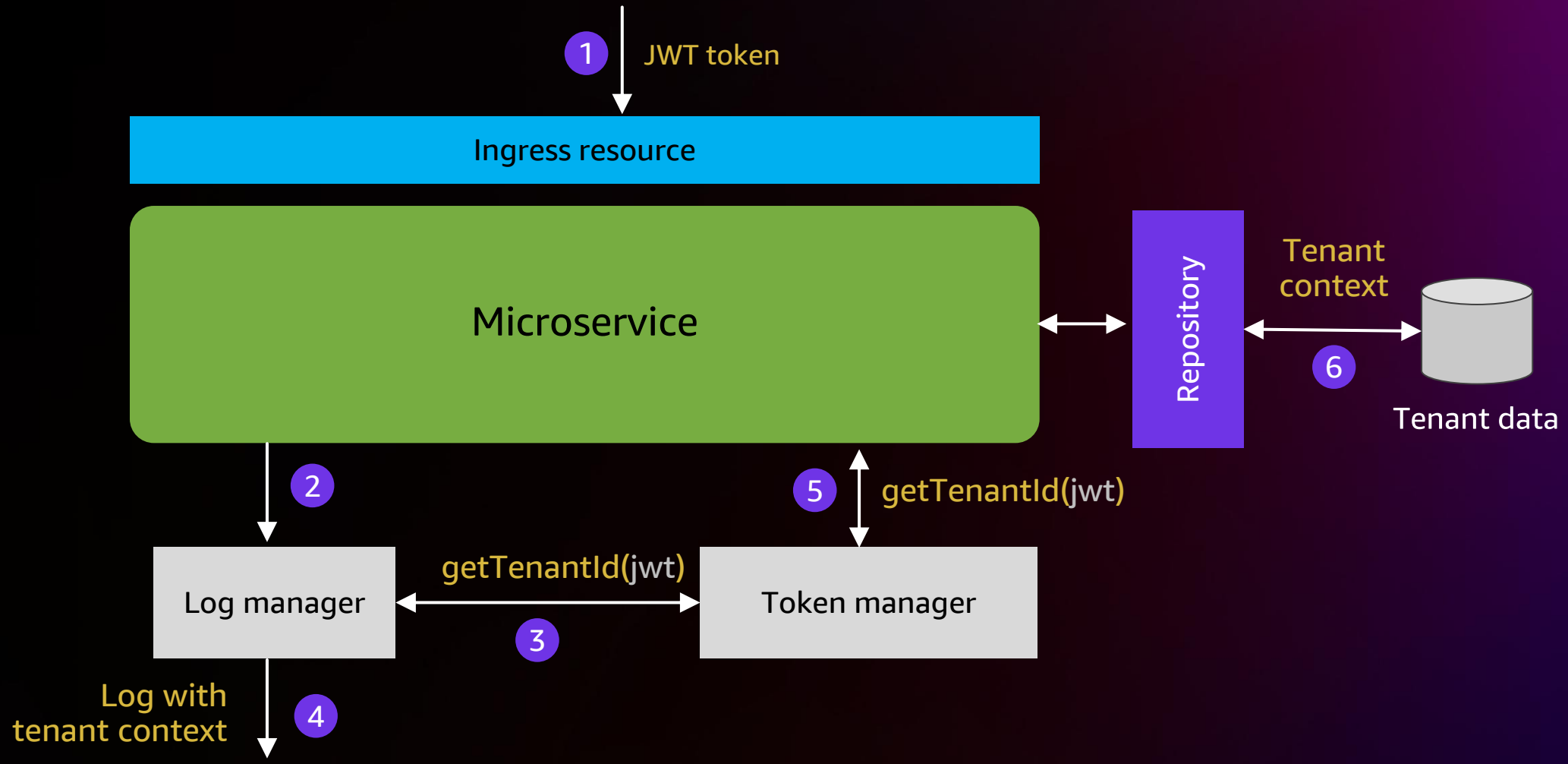
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Let the modernization begin



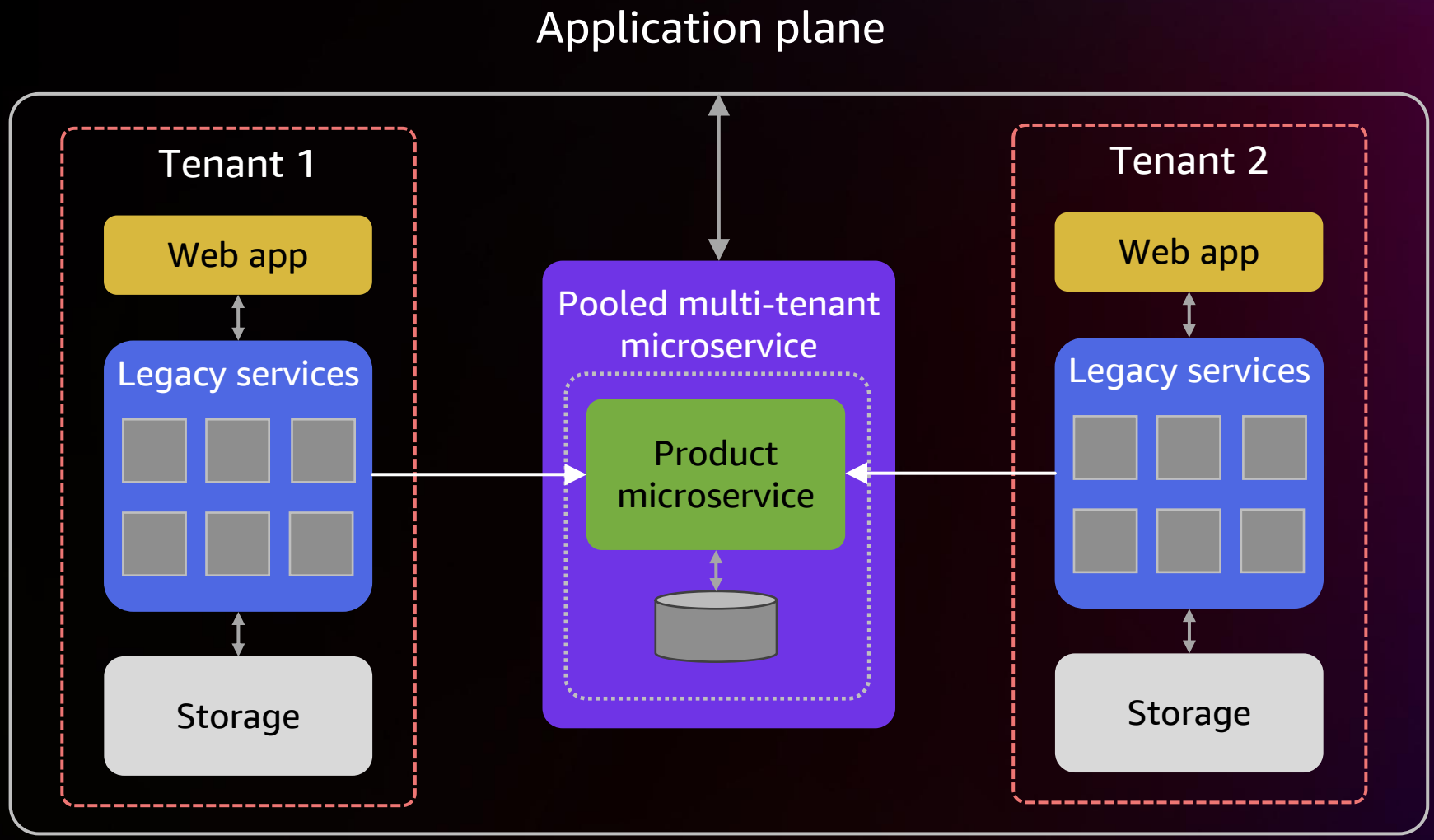
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Creating a modernized multi-tenant microservice



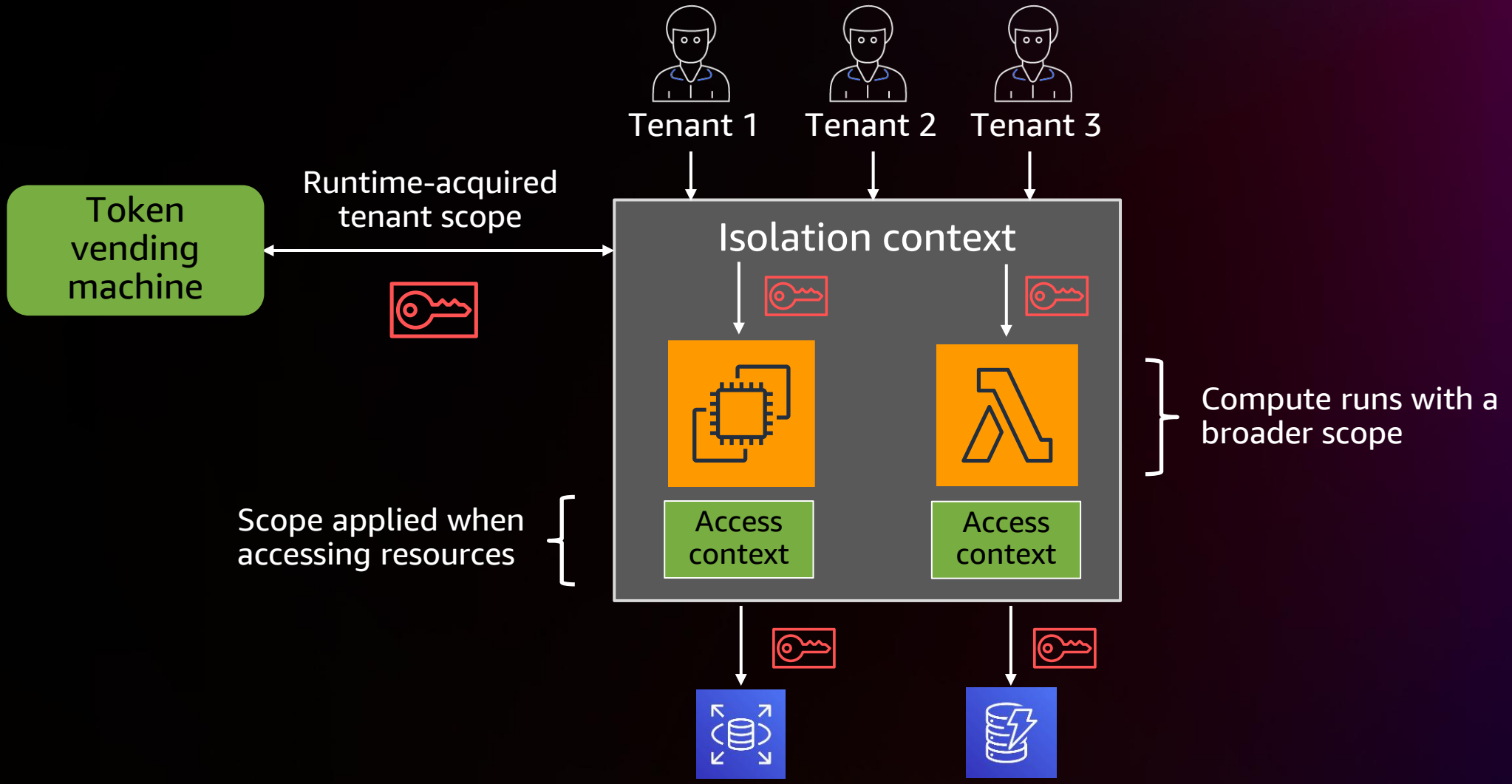
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Extending the model: Pooled services



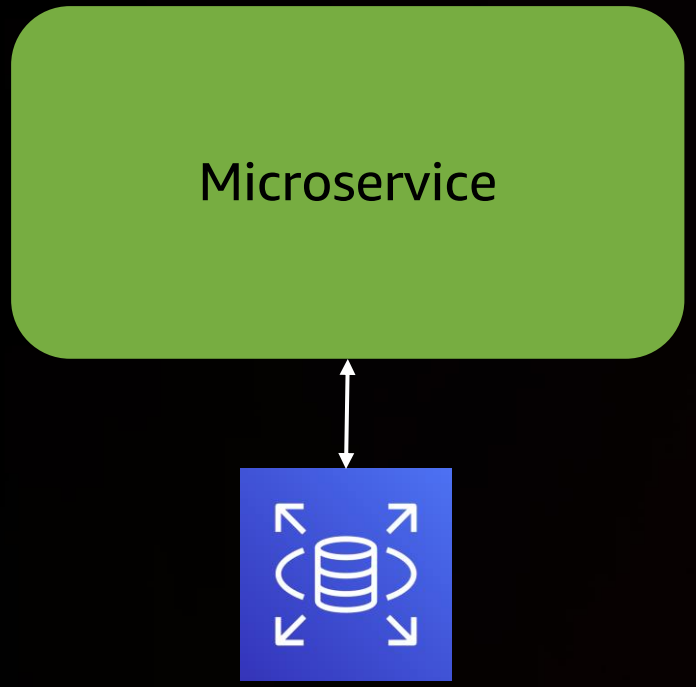
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

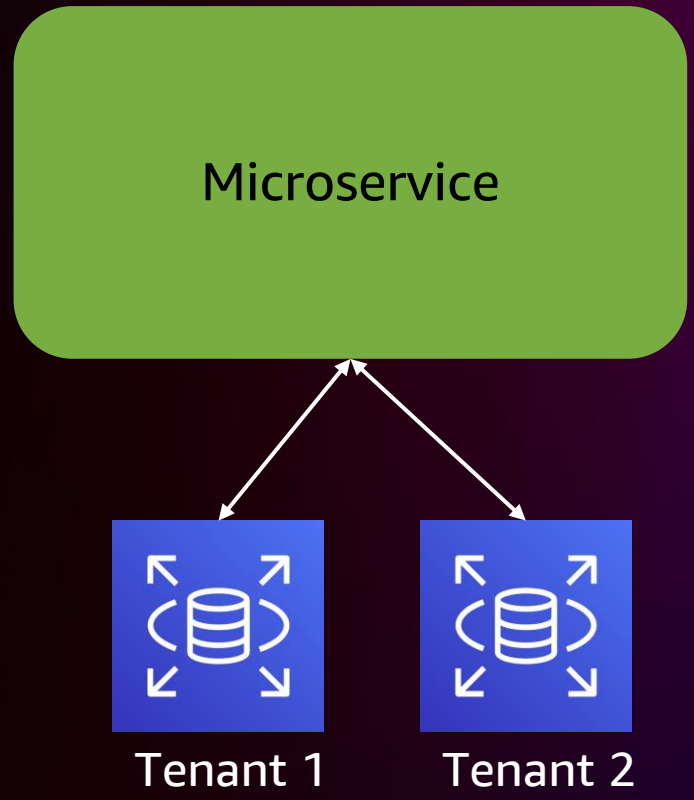
Adding isolation support



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Migrating your microservice storage





aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

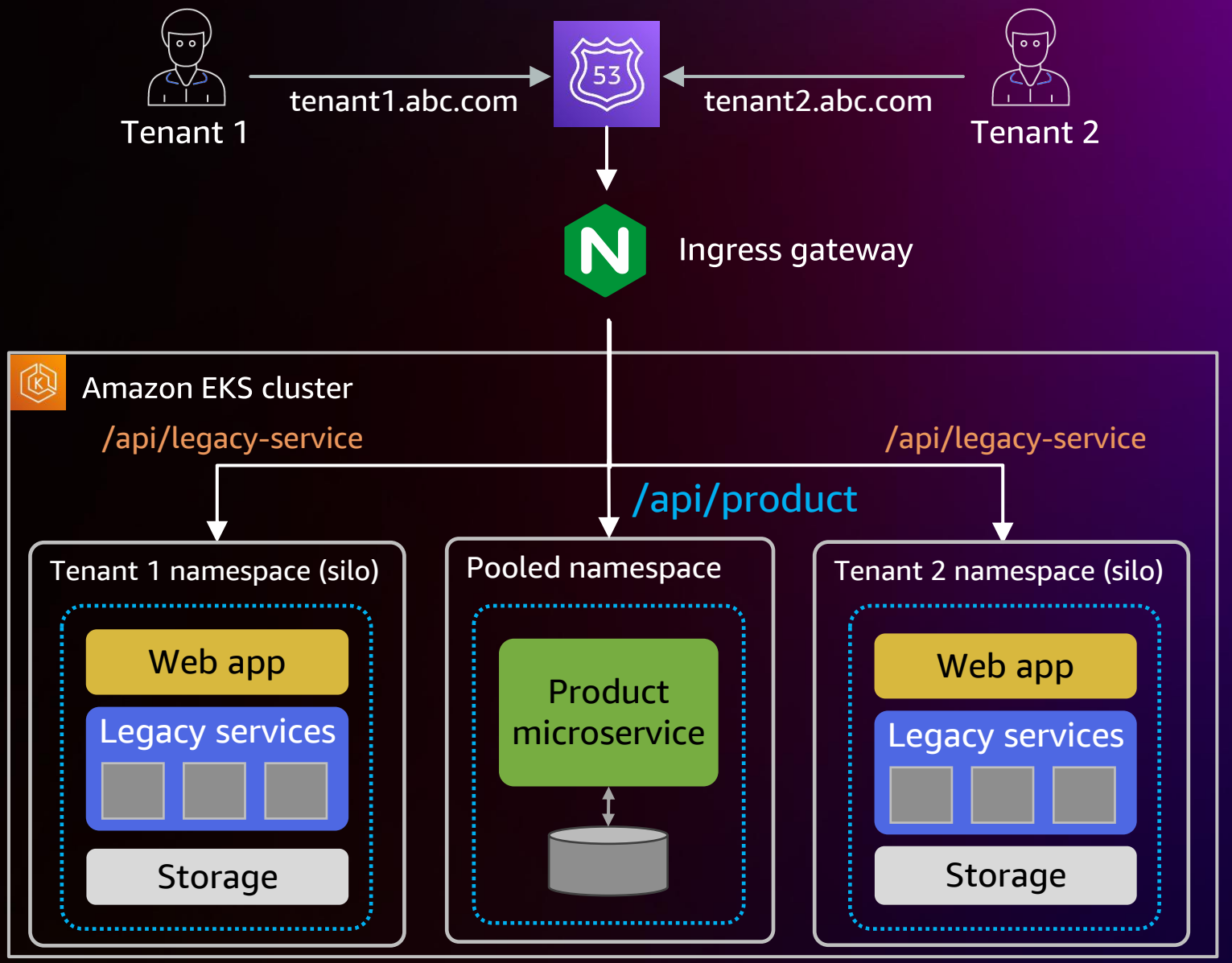
Updating our routing model

monolith.yaml

```
- host: tenant1.abc.com
  http:
    paths:
      - path: /api/legacy-service
        backend:
          service:
            name: legacy-service
            port:
              number: 80
```

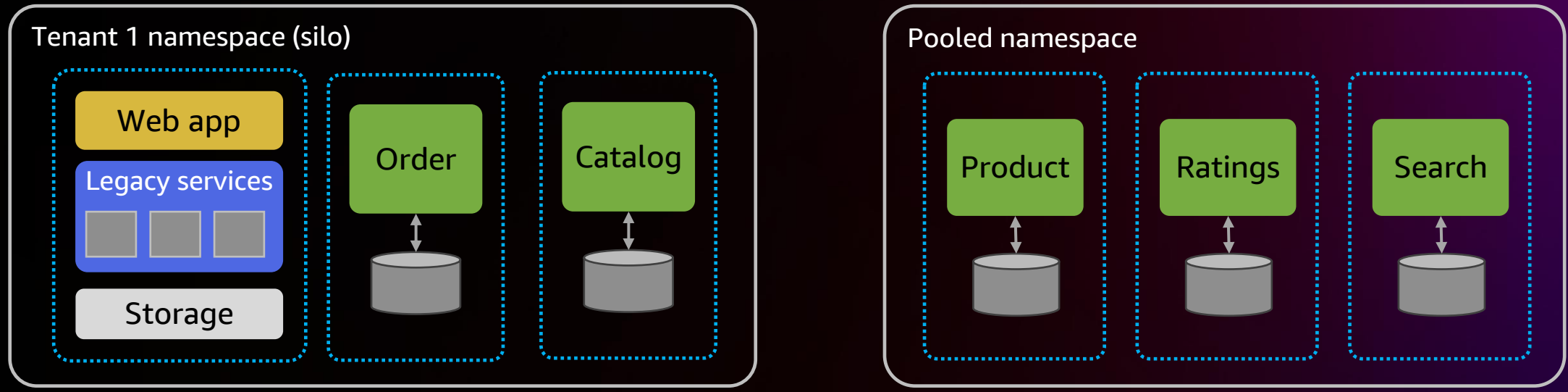
product-service.yaml

```
- host: *.abc.com
  http:
    paths:
      - path: /api/product
        backend:
          service:
            name: product-service
            port:
              number: 80
```



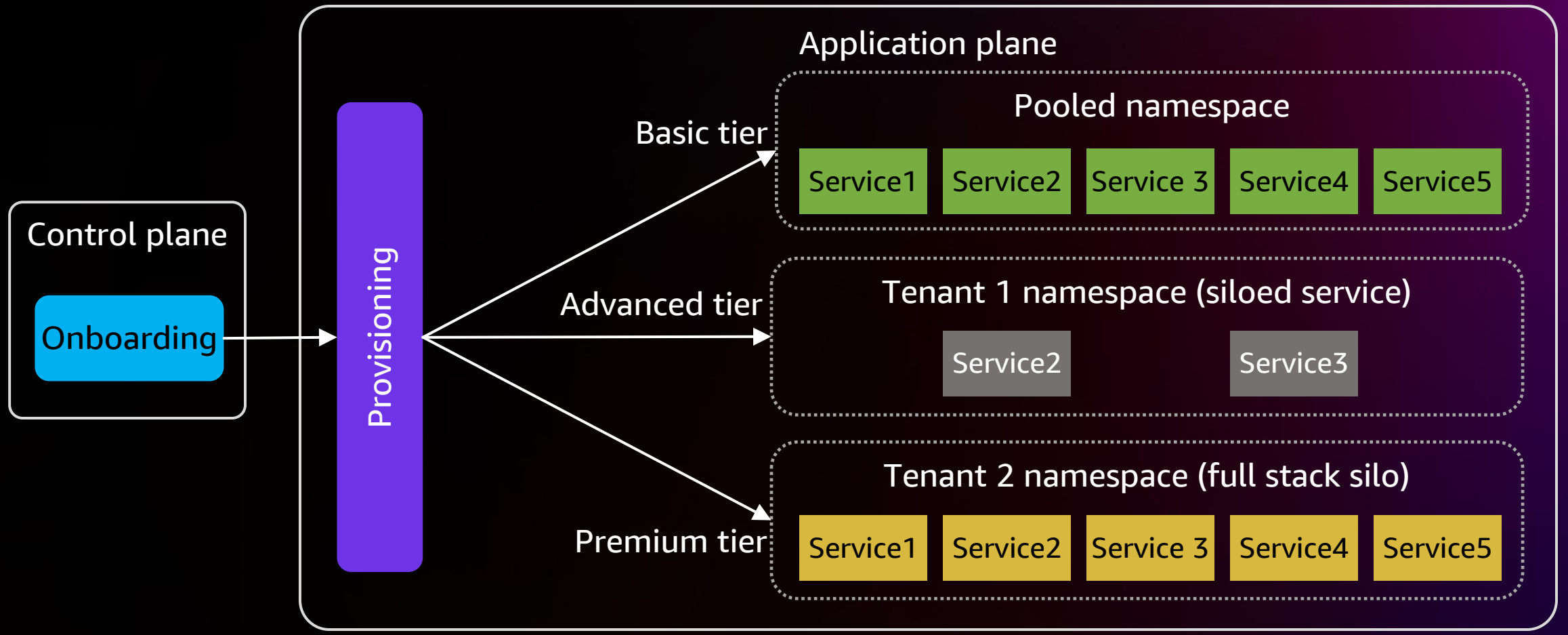
aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Supporting silo and pool during migration



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Provisioning: An ongoing effort

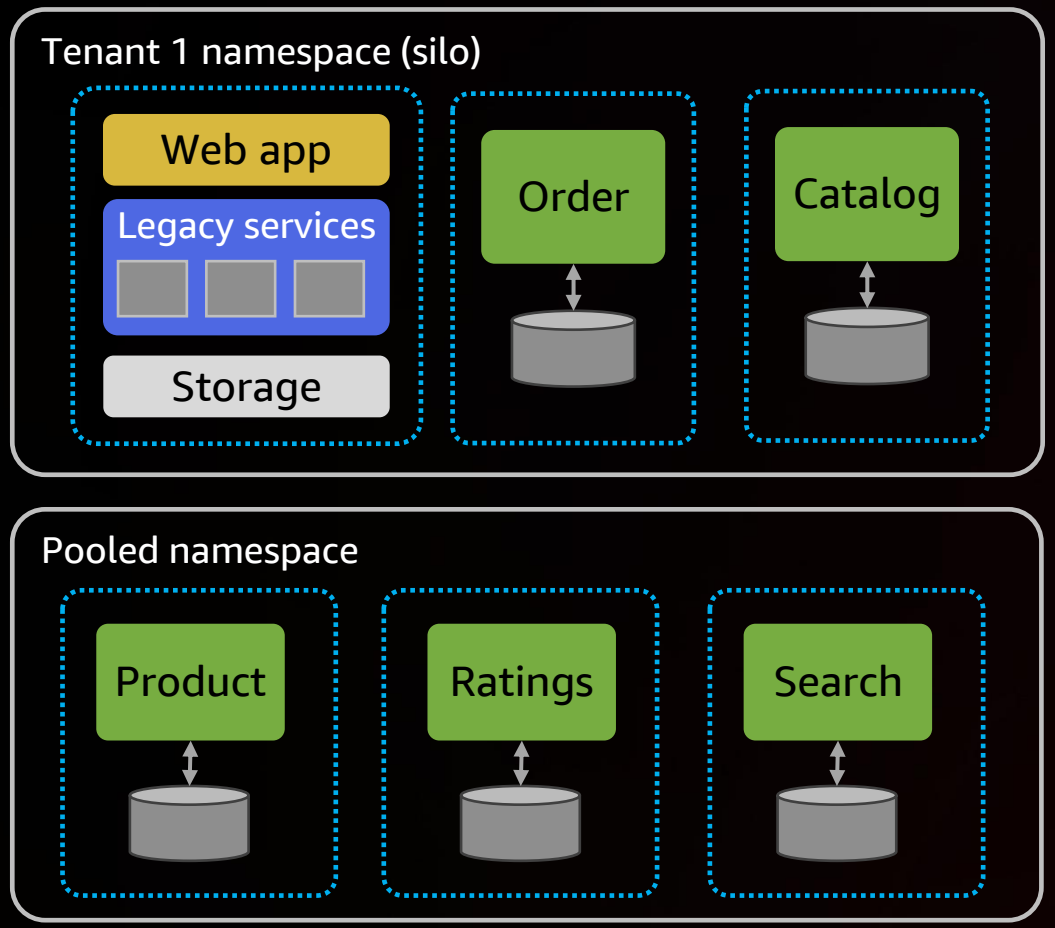


More capabilities = more provisioning/configuration complexity

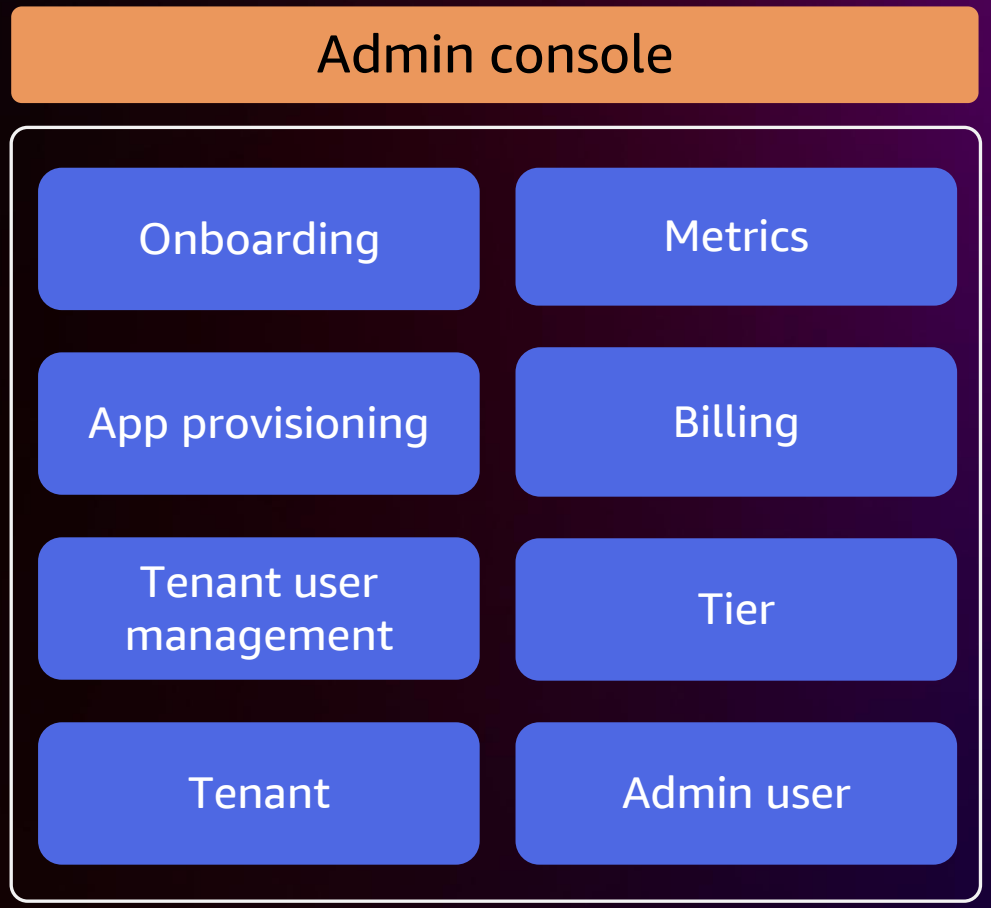
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

All the pieces are in place

Application plane



Control plane



aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Takeaways

- Think incrementally – get to SaaS as fast as you can
- Start with the control plane
- Make automated onboarding and tenant-aware operations a priority
- Expect to refactor aspects of your legacy solution
- Embrace full stack silo as a starting point
- Expect the first steps of microservice modernization to be hard
- Refactoring is not likely a one-time exercise – allow yourself to get it wrong
- Set a high bar for new/refactored microservices

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

More SaaS sessions

Breakout sessions

- SAS305 – SaaS architecture patterns: From concept to implementation
- SAS405 – SaaS microservices deep dive: Simplifying multi-tenant development
- 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

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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

Builders' session

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

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

Tod Golding
todg@amazon.com



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

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.