
KubeCon
CloudNativeCon
North America 2022
BUILDING FOR THE ROAD AHEAD
DETROIT 2022

KubeCon | CloudNativeCon
North America 2022

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

Like Peas and Carrots:

Argo CD and Crossplane for Infrastructure Management

Jesse Suen, Akuity
Victor Farci, Upbound

KubeCon | CloudNativeCon

North America 2022

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

October 24-28, 2021

Jesse Suen

CTO

Akuity

Victor Farcic

Developer Advocate

Upbound

KubeCon | CloudNativeCon

North America 2022

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

How We Got Here?

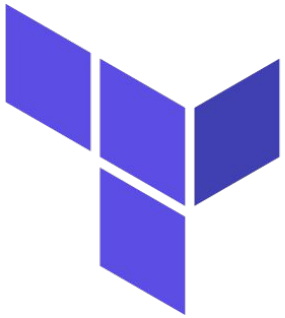
Configuration Management

Mutable - Baremetal

Infrastructure as Code (IaC)

AWS CLOUDFORMATION

Pulumi



HashiCorp

Terraform

Immutable - Reactive

Control Planes - Now

- API
- CRDs
- Continuous drift-detection and reconciliation
- Ecosystem

Custom Resource Definitions (CRDs)

```yaml
apiVersion: ec2.aws.crossplane.io/v1alpha1
kind: Instance
metadata:
  name: dot
  labels:
    app: dot
spec:
  forProvider:
    imageId: ami-052efd3df9dad4825
    region: us-east-1
    instanceType: t2.micro
    subnetIdSelector:
      matchLabels:
        app: dot
```

Crossplane

Composite Resources (XRs)

```yaml
apiVersion: devopstoolkitseries.com/v1alpha1
kind: ClusterClaim
metadata:
  name: a-team-eks
spec:
  id: a-team-eks
  compositionSelector:
    matchLabels:
      provider: aws
      cluster: eks
  parameters:
    nodeSize: medium
    minNodeCount: 3
  writeConnectionSecretToRef:
    name: a-team-eks
```

Crossplane

Internal Developer Platforms (IDPs)

Crossplane

Case Study: Akuity

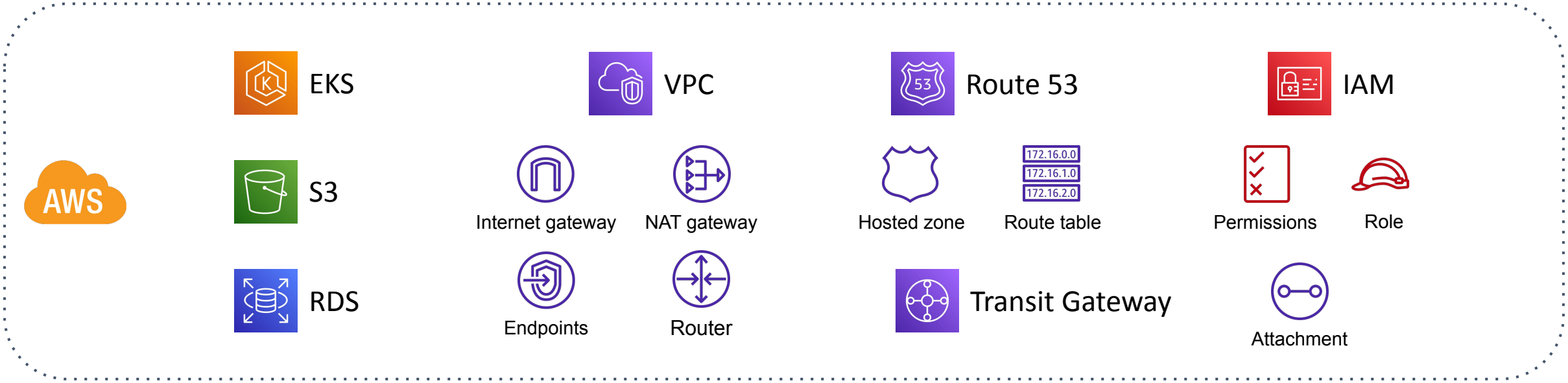
Why Crossplane?

- Manage infrastructure the same way as our apps (Argo CD)
- Coordination between infra and apps (IAM)
- Want standardized, self-service infra with simplified knobs
- Want to treat our clusters and infra as cattle

Akuity: How we use Crossplane

Providers:

- aws-provider
- helm-provider
- kubernetes-provider



Akuity Compositions

EKS

Cluster
Roles
Policies
OIDCProvider
NodeGroup
KMS Key
KMS Alias
Addons
AkuityAddons

IRSA

Policy
Role
RolePolicyAttachment

RDS

DBCluster
DBInstance
DBSubnetGroup
SecurityGroup
KMS Key
KMS Alias
DBClusterParameterGroup
DBParameterGroup

EKSNetwork

VPC
Gateway
InternetGateway
3 Public Subnets
3 Private Subnets
3 NAT IPs
3 NAT Gateways
Routes
RouteTables

RDSNetwork

VPC
3 Private Subnets
RouteTables

Akuity Addon Compositions

Karpenter

IRSA
InstanceProfile
Helm Release
Provisioner

Cert-Manager

IRSA
Namespace
ServiceAccount

External Secrets

IRSA
ClusterSecretStore
Helm Release

AWS Load Balancer Controller

IRSA
Helm Release

AWS Distro Open Telemetry (ADOT)

IRSA
Namespace
ServiceAccount

External DNS

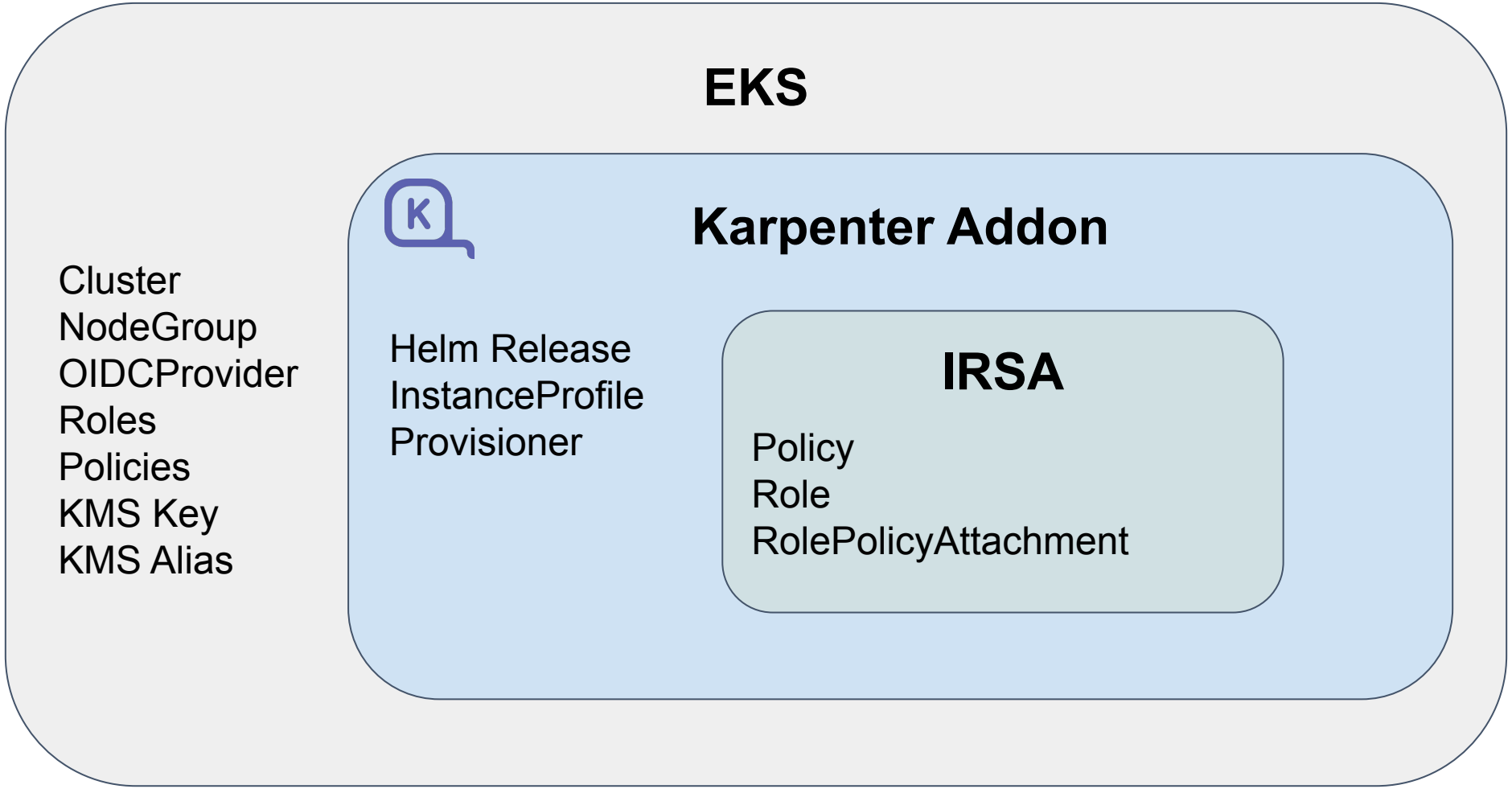
IRSA
Namespace
ConfigMap
ServiceAccount

Karpenter EKS Addon

Reduced knobs for simplified experience

```yaml
apiVersion: addons.akuity.io/v1alpha1
kind: Karpenter
metadata:
  name: mycluster
spec:
  capacityType: spot
  instanceTypes: [m5.large, m5.xlarge, c5.large, c5.xlarge]
  limits:
    cpu: "100"
   ...
```

Akuity EKS Addon Compositions



EKS - IRSA

Challenge: Cross referencing between AWS / Kubernetes

IAM Role - arn:aws:iam::123:role/eks.cluster-foo.aws-lbc

Role Policy Attachment

IAM Policy

```
{
 "Version": "2012-10-17",
 "Statement": [{
   "Effect": "Allow",
   "Action": [
    "ec2:DescribeAddresses",
    "ec2:DescribeAvailabilityZones",
    "ec2:DescribeInternetGateways",
     ...
  ]}]
}
```

```
{
 "Version": "2012-10-17",
 "Statement": [{
     "Effect": "Allow",
     "Principal": {
       "Federated":
         "arn:aws:iam::123:oidc-provider/oidc.eks.usw2.amazonaws.com/id/ABC123"
     },
     "Action": "sts:AssumeRoleWithWebIdentity",
     "Condition": {
       "StringLike": {
         "oidc.eks.usw2.amazonaws.com/id/ABC123:sub":
                 "system:serviceaccount:aws-lbc:aws-lbc"
```

ServiceAccount

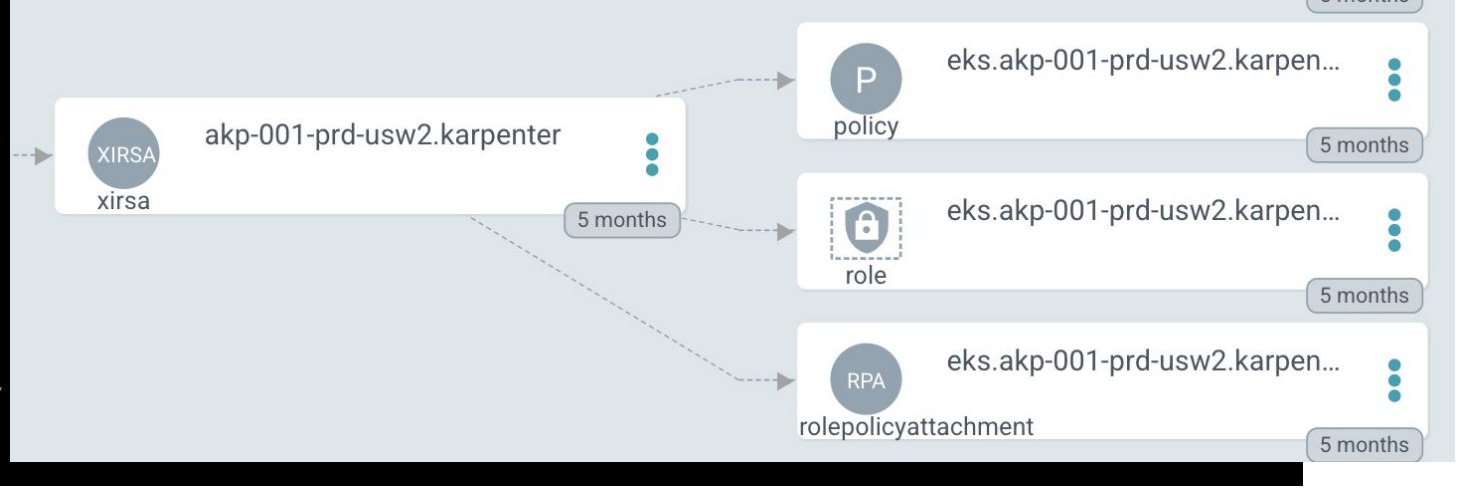
```
apiVersion: v1
kind: ServiceAccount
metadata:
  annotations:
    eks.amazonaws.com/role-arn: |-
      arn:aws:iam::123:role/eks.cluster-foo.aws-lbc
  name: aws-lbc
  namespace: aws-lbc
```

cluster-foo

oidc.eks.usw2.amazonaws.com/id/ABC123

EKS - IRSA Composition

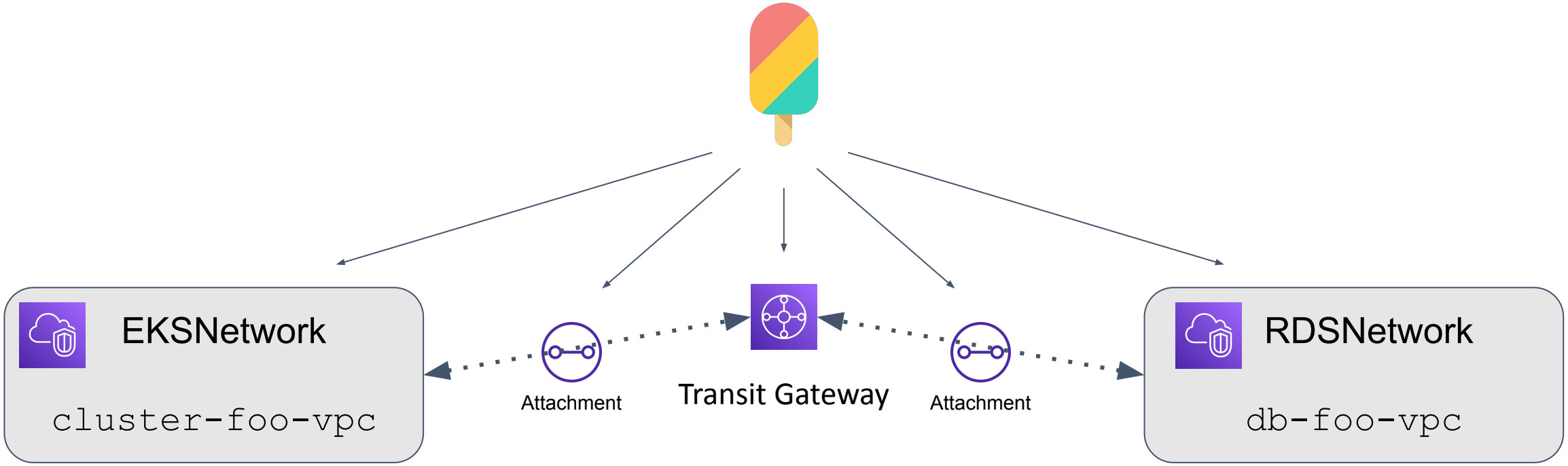
```
apiVersion: aws.akuity.io/v1alpha1
kind: IRSA
metadata:
 name: cluster-foo.aws-lbc
spec:
 serviceAccountName: aws-lbc
 namespace: aws-lbc
 clusterRef:
   endpoint: https://ABCD1234.gr7.usw2.eks.amazonaws.com
   id: cluster-foo
   oidc:
     provider: oidc.eks.usw2.amazonaws.com/id/ABCD1234
     providerARN: arn:aws:iam::123:oidc-provider/oidc.eks.usw2.amazonaws.com/id/ABCD1234
 policyDocument: |
   {
    "Version": "2012-10-17",
    "Statement": [
     {
      "Effect": "Allow",
      "Action": [
       "ec2:DescribeAddresses",
       "ec2:DescribeAvailabilityZones",
       ...
```



Managed Resources

Also deploy managed resources directly (not in a Composition):

- e.g. Route53, Routes, TransitGateways, Bespoke IAM



Crossplane vs. Argo CD for Addons

Pros:

- Cluster works out of the box with no extra steps
- IAM can be coordinated

Cons:

- Cumbersome addons upgrades: require a new composition version
- All clusters updated simultaneously

Pros:

- GitOps-based update process
- Finer control of versions in envs
- Easy to modify ConfigMap/Secrets
- Addons can be made optional

Cons:

- Harder to coordinate with IAM
- Extra step after cluster creation

Crossplane vs. Argo CD for Addons

Solution: Use Crossplane for some addons, Argo CD for others

Type	Description	Managed by
Critical	Essential for cluster to function (autoscaling). Fully managed by Crossplane.	
IAM Only	***Optional*** addons requiring AWS resources. IAM managed by Crossplane. Deployment/Config managed by Argo CD.	 
Other	No dependency to AWS. Normal applications, fully managed by Argo CD	

Crossplane vs. Argo CD for Addons

Crossplane vs. Argo CD for Kubernetes Resources

Critical

- Karpenter
- AWS Load Balancer
- External Secrets

IAM Only (Optional)

- Cert Manager
- External DNS
- ADOT

The Rest

- Metrics Server
- Calico
- Traefik
- Prometheus

Best Practice: Controlling Blast Radius

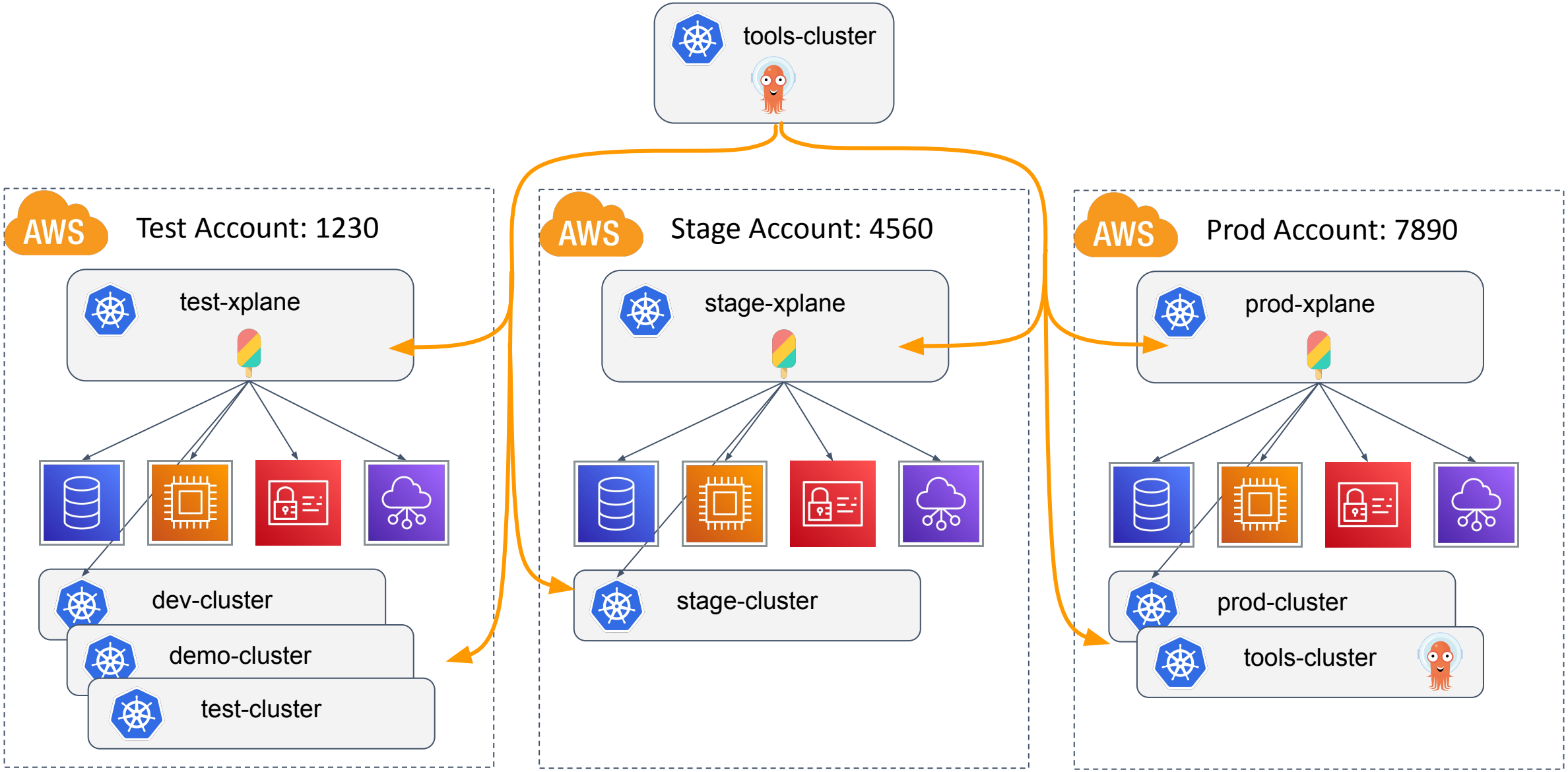
Problem:

- Rolling out new Compositions, Providers versions is risky
- Bugs in your composition affects all instances simultaneously

Solution:

- Run multiple Crossplane instances (per environment)
- Promote Compositions progressively to each environment

Best Practice: Crossplane per Env



Tips & Tricks

Argo CD: Pruning

Problem:

- Argo CD wants to prune the XRs produced by XRCs

Solution:

- Use “annotation” based resource tracking

```
kind: ConfigMap
metadata:
 name: argocd-cm
data:
 application.resourceTrackingMethod: annotation
```

- Requires Argo CD v2.4.9 (#8683)

Argo CD: Performance Tuning

Problem:

- Crossplane installs hundreds of CRDs
- You're probably not using most of them
- Argo CD list/watches each CRD, even if it's not used
- Increased K8s API and app-controller memory pressure

Argo CD: Resource Exclusions

Hundreds of open connections to K8s API

list + watch

```
certificates.acm.aws.crossplane.io
apikeys.apigateway.aws.crossplane.io
authorizers.apigateway.aws.crossplane.io
deployments.apigateway.aws.crossplane.io
documentationparts.apigateway.aws.crossplane.io
documentationversions.apigateway.aws.crossplane.io
addresses.ec2.aws.crossplane.io
flowlogs.ec2.aws.crossplane.io
instances.ec2.aws.crossplane.io
internetgateways.ec2.aws.crossplane.io
launchtemplates.ec2.aws.crossplane.io
classifiers.glue.aws.crossplane.io
connections.glue.aws.crossplane.io
crawlers.glue.aws.crossplane.io
accesskeys.iam.aws.crossplane.io
grouppolicyattachments.iam.aws.crossplane.io
groups.iam.aws.crossplane.io
...
```

Argo CD: Resource Exclusions

Reduce connections by excluding unused resources

```yaml
kind: ConfigMap
metadata:
 name: argocd-cm
data:
 resource.exclusions: |
  - apiGroups:
    - apigateway.aws.crossplane.io
    - apigatewayv2.aws.crossplane.io
    - glue.aws.crossplane.io
    - cache.aws.crossplane.io
    - kafka.aws.crossplane.io
    - dax.aws.crossplane.io
    - docdb.aws.crossplane.io
    - dynamodb.aws.crossplane.io
...
```

certificates.acm.aws.crossplane.io
~~apikeys.apigateway.aws.crossplane.io~~
~~authorizers.apigateway.aws.crossplane.io~~
~~deployments.apigateway.aws.crossplane.io~~
~~documentationparts.apigateway.aws.crossplane.io~~
~~documentationversions.apigateway.aws.crossplane.io~~
addresses.ec2.aws.crossplane.io
flowlogs.ec2.aws.crossplane.io
instances.ec2.aws.crossplane.io
internetgateways.ec2.aws.crossplane.io
launchtemplates.ec2.aws.crossplane.io
~~classifiers.glue.aws.crossplane.io~~
~~connections.glue.aws.crossplane.io~~
~~crawlers.glue.aws.crossplane.io~~
accesskeys.iam.aws.crossplane.io
grouppolicyattachments.iam.aws.crossplane.io
groups.iam.aws.crossplane.io
...

Argo CD: Resource Exclusions

Declutter Argo CD UI by excluding ProviderConfigUsages (Crossplane implementation detail)

```yaml
kind: ConfigMap
metadata:
 name: argocd-cm
data:
 resource.exclusions: |
   - kinds:
     - ProviderConfigUsage
     apiGroups:
     - "*"
```

Argo CD: Performance Tuning

Problem:

- Crossplane installs hundreds of CRDs
- API Discovery of those CRDs get throttled (#3272)
- Currently, Argo CD Discovery is not efficient (#448)

(Temporary) Workaround:

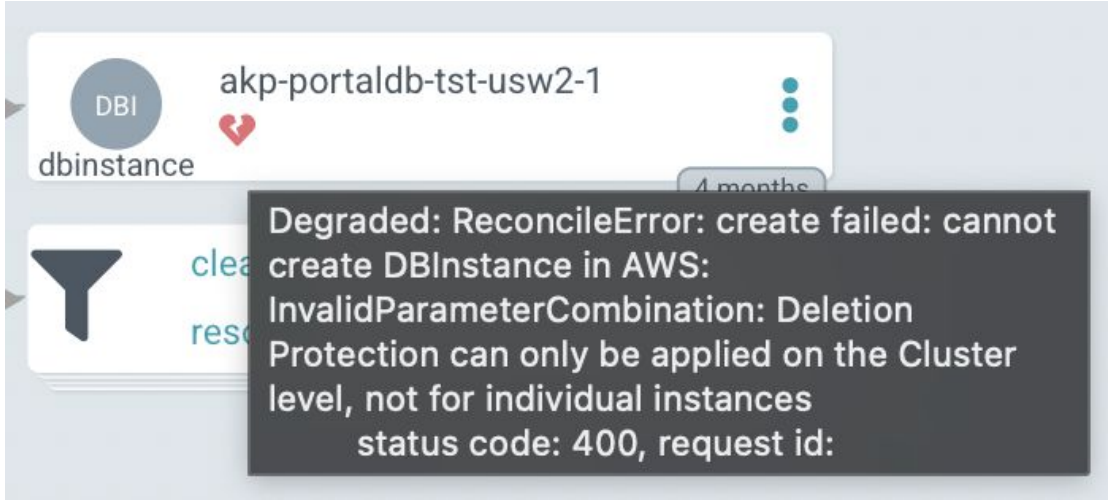
- Increase Kubernetes QPS limit to app-controller

```
env:
- name: ARGOCD_K8S_CLIENT_QPS
  value: "300"
```

Argo CD: Health Checks

Tip & Trick:

- Write Argo CD health check for Crossplane resources

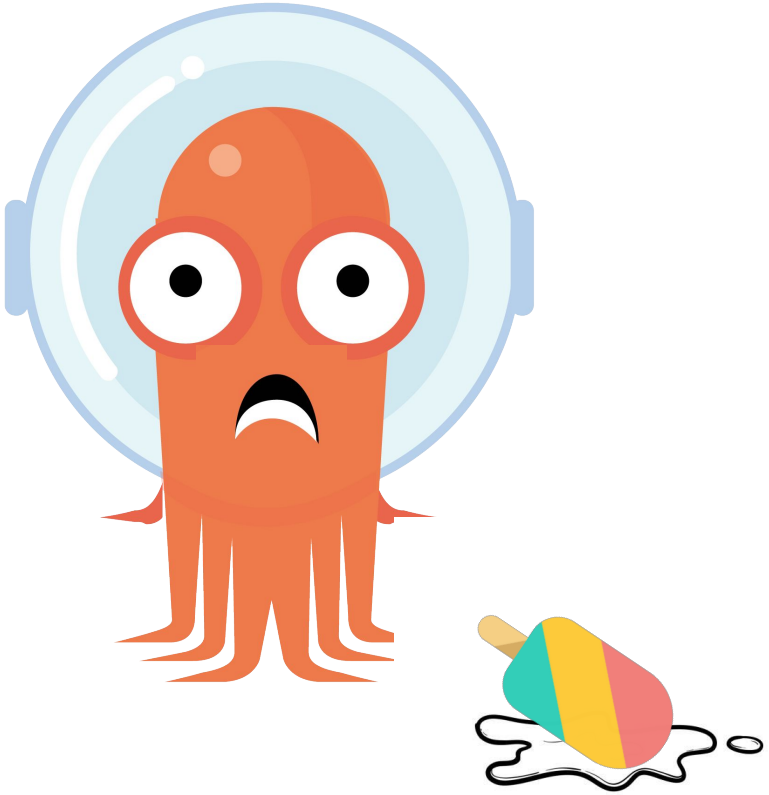



```yaml
status:
 conditions:
 - lastTransitionTime: "2022-10-14T23:53:45Z"
   message: "create failed: cannot create DBInstance in AWS: InvalidParameterCombination:
     Deletion Protection can only be applied on the Cluster level, not for individual
     instances\n\tstatus code: 400, request id: "
   reason: ReconcileError
   status: "False"
   type: Synced
```

Argo CD: Health Checks

```
kind: ConfigMap
metadata:
 name: argocd-cm
data:
 resource.customizations.health.rds.aws.crossplane.io_DBInstance: |
   hs = {}
   hs.status = "Healthy"
   synced = true
   for i, condition in ipairs(obj.status.conditions) do
       if condition.type == "Synced" and condition.status ~= "True"
           hs.status = "Degraded"
           synced_message = condition.reason
           if condition.message then
               synced_message = synced_message .. ": " ..condition.message
           end
       end
   end
   return hs
```

Future Improvement: Shared Resource Health Checks (#4212)



Challenges

Crossplane Challenges

Adopting existing cloud resources

- Many AWS resources receive random IDs
  - e.g. VPCs, Subnets, SecurityGroups, HostedZones
- Crossplane will recreate them if XR is recreated
- `external-name` annotation helps, but breaks GitOps UX

```
apiVersion: ec2.aws.crossplane.io/v1beta1
kind: VPC
metadata:
 annotations:
   crossplane.io/external-name: vpc-0a1b2c3d4e5f6789
 name: foo-vpc
```

Tip: Use formulated XR names (avoid generateName)!

Crossplane Limitations / Wishlist

Conditional Resources (#2712)

- In a Composition, I want to create resources depending on the values of input parameters.

Use Cases:

- Enable/Disable Cluster Addons
- Prod vs. Dev Databases

Current Workaround:

- Violate DRY. We resort to code generation to create variations of compositions (RDS, RDSDev)

Crossplane Limitations / Wishlist

Cross-Resource Referencing (#1770)

- In a Composition, I would like to reference fields of *another* resource as input to the current resource

Use Cases:

- Enables loosely coupled compositions
- Simplify deploy process

Current Workaround: Staged deploys

1. Deploy two VPCs
2. Extract VPC ID of each
3. Copy & paste VPC ID into TransitGateway Attachments

Future Improvements

Cheaper Control Planes

- Currently running dedicated EKS clusters just for Crossplane
- Multiple accounts/environments == expensive!
- Instead of EKS as our crossplane cluster, K3s in EKS

Argo CD + Crossplane

Thank You!

Please scan the QR Code above to leave feedback on this session

KubeCon | CloudNativeCon
North America 2022

BUILDING FOR THE ROAD AHEAD

DETROIT 2022