CNI-Chaining with Cilium on EKS: the good, the bad and how to break the chains quietly

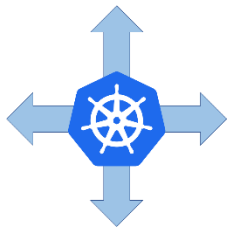
Nathanael, Liechti

Cloud-Native Bern Meetup, 30st May 2024

Quiz

- Is there anything Cilium can't do?
- What does EKS stand for?
- In which (minor) version of the CNI-spec was CNI-chaining support added?

EKS @ Swiss Post

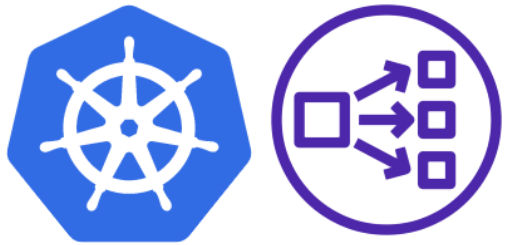


29 clusters

300 nodes

10000 pods

700 apps



A Kubernetes controller for Elastic Load Balancers

AWS VPC-CNI

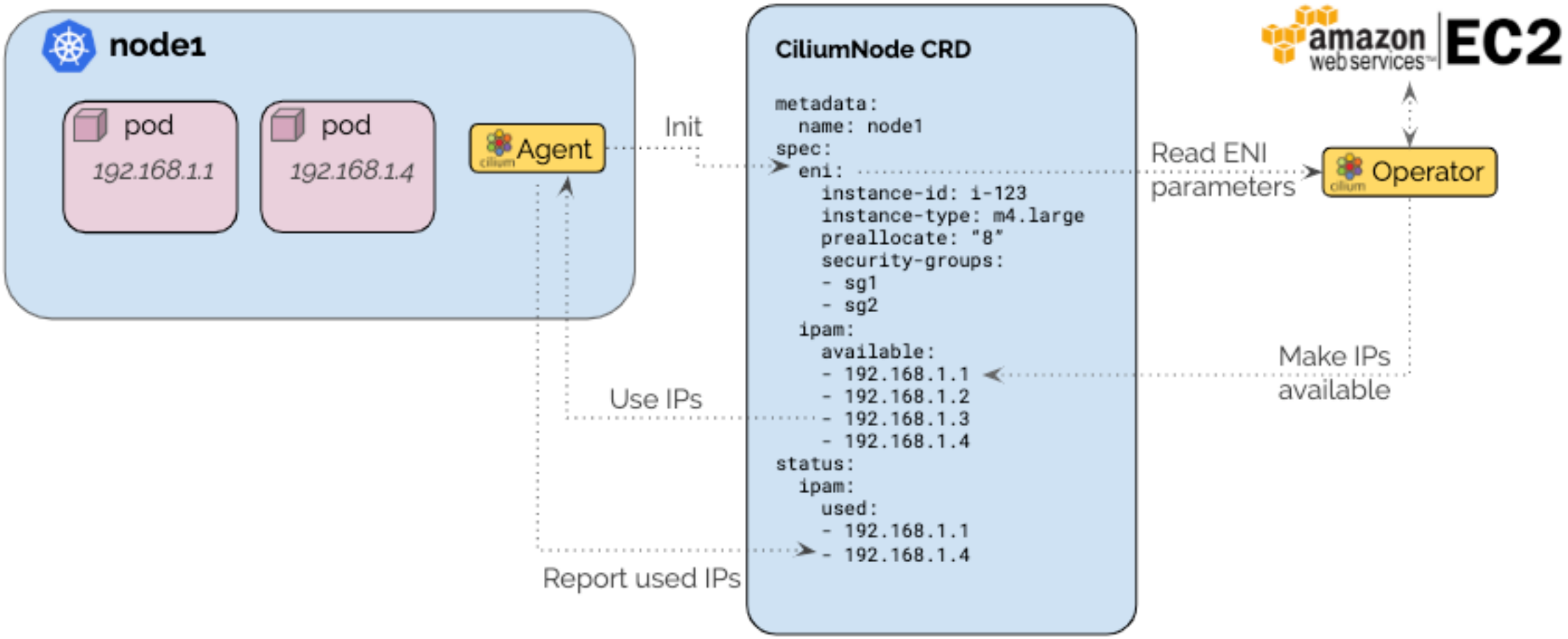
VPC-wide routable addresses for pods

Default CNI-plugin on EKS

Managed as EKS addon

Support for network policies (since v1.14)

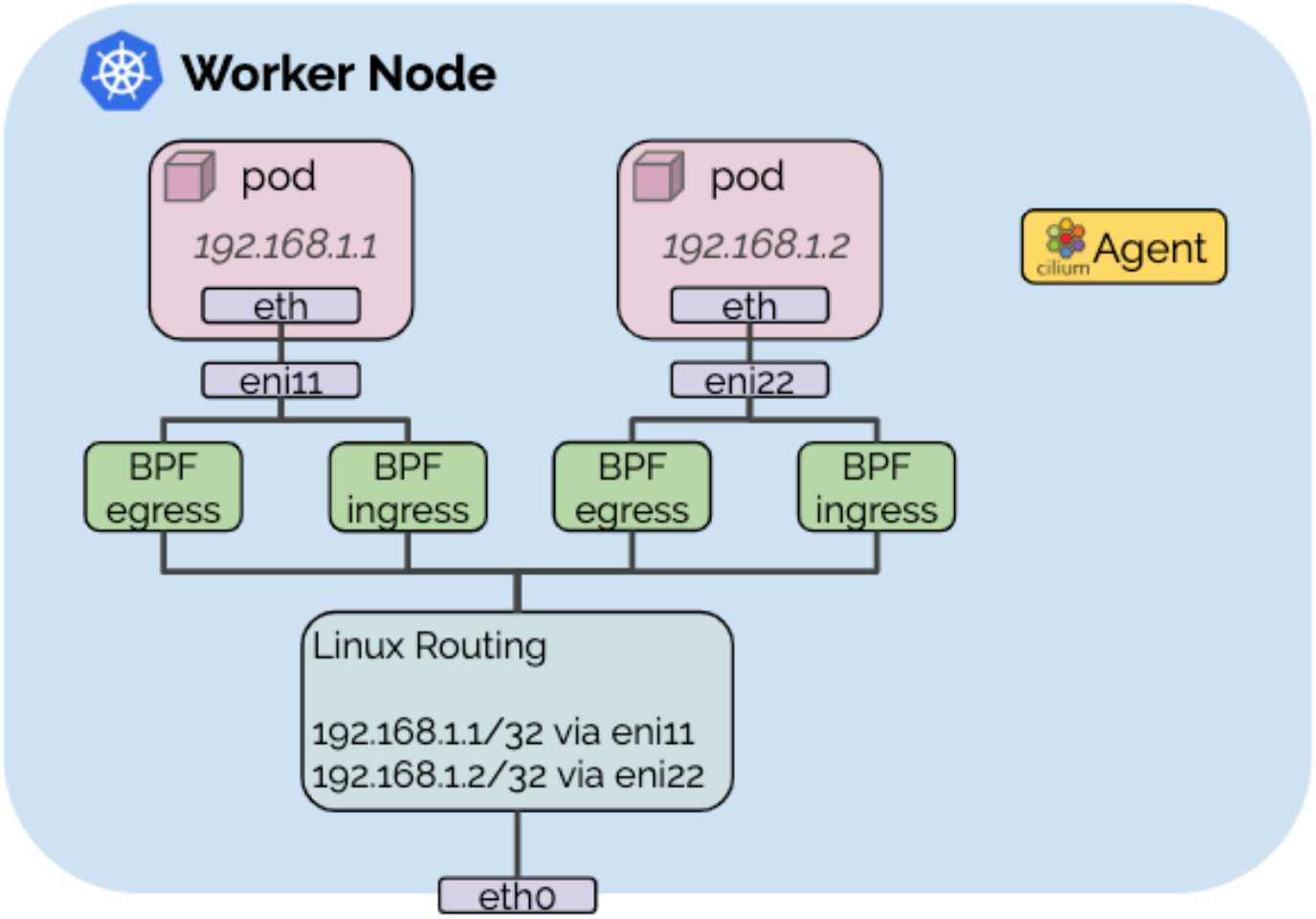
Cilium ENI mode



CNI-chaining?


```
{
  "cniVersion": "0.3.1",
  "name": "aws-cni",
  "plugins": [
     {
       "name": "aws-cni",
       "type": "aws-cni",
       "vethPrefix": "eni",
       "mtu": "9001",
       "pluginLogFile": "/var/log/aws-routed-eni/plugin.log",
       "pluginLogLevel": "DEBUG"
     },
     {
       "name": "cilium",
       "type": "cilium-cni",
       "enable-debug": false,
       "log-file": "/var/run/cilium/cilium-cni.log"
     }
  ]
}
```

CNI-chaining responsibilities

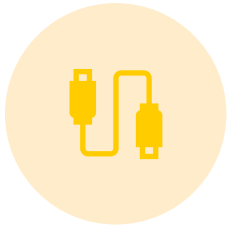


AWS-CNI:
- *Device plumbing*
- *IPAM (ENI)*
- *Routing*

Cilium
- *Load-balancing*
- *Network policy*
- *Encryption*
- *Multi-cluster*
- *Visibility*

https://docs.cilium.io/en/stable/installation/cni-chaining/

Why CNI-chaining?



Easy installation (plug on top of existing vpc-cni)

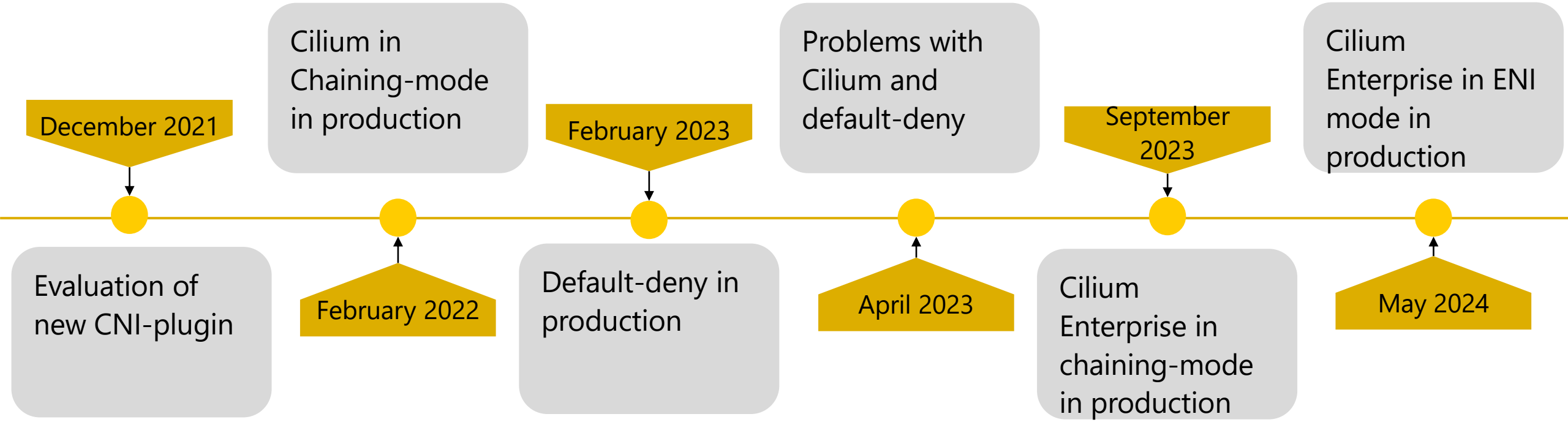
No official toggle for vpc-cni containers-roadmap#923

IPv6 support cilium#18405

Security Groups for pods

Support from AWS for vpc-cni (if you are not using Cilium Enterprise)

Our CNI-journey



CNI-chaining ⚡⚡⚡

Note

Some advanced Cilium features may be limited when chaining with other CNI plugins, such as:

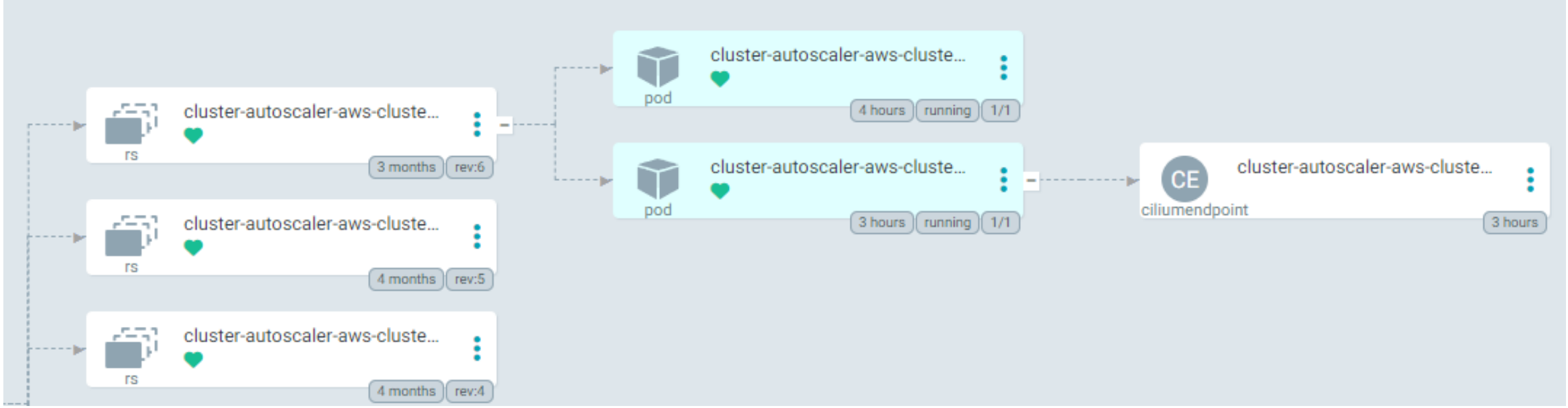
- Layer 7 Policy (see GitHub issue 12454)
- IPsec Transparent Encryption (see GitHub issue 15596)

Video

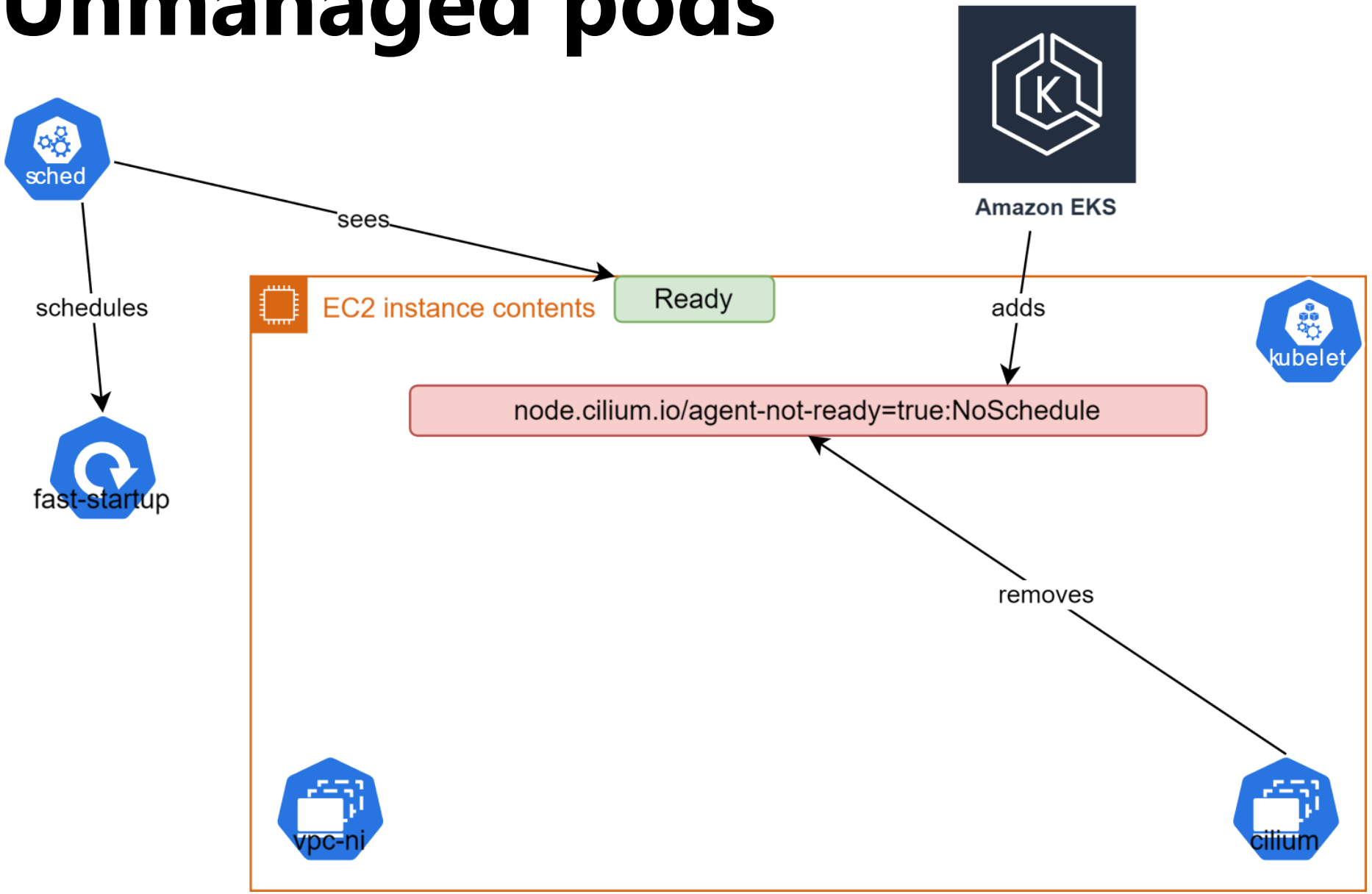
If you require advanced features of Cilium, consider migrating fully to Cilium. To help you with the process, you can watch two Principal Engineers at Meltwater talk about how they migrated Meltwater's production Kubernetes clusters - from the AWS VPC CNI plugin to Cilium.

Unmanaged pods

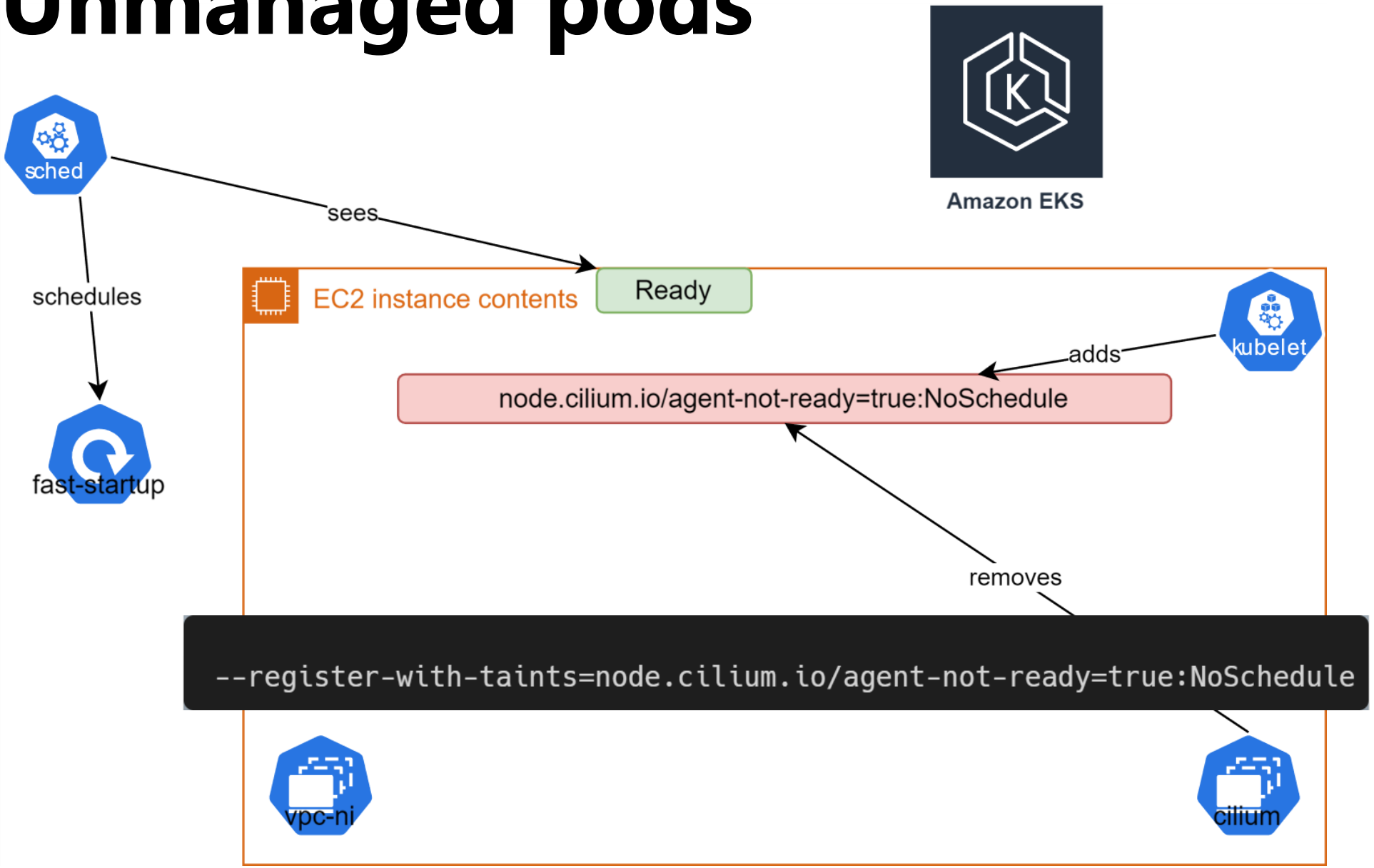
Unmanaged pods are pods managed by vpc-cni but not by Cilium



Unmanaged pods



Unmanaged pods



CNI-chaining problems

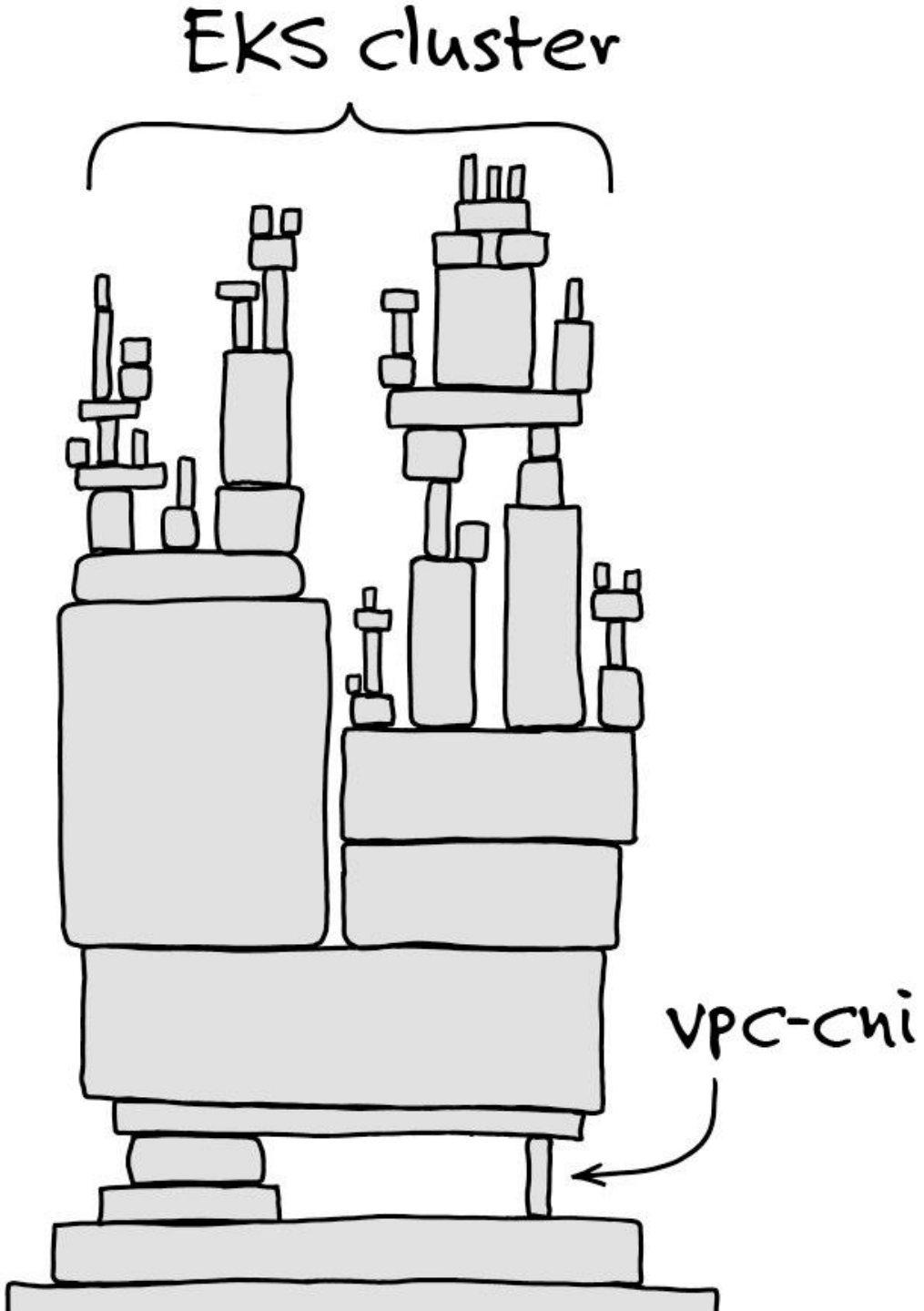
Unmanaged pods (unlikely but still possible)

CNI-Chaining feature incompatibility

Network Policy implementation of vpc-cni

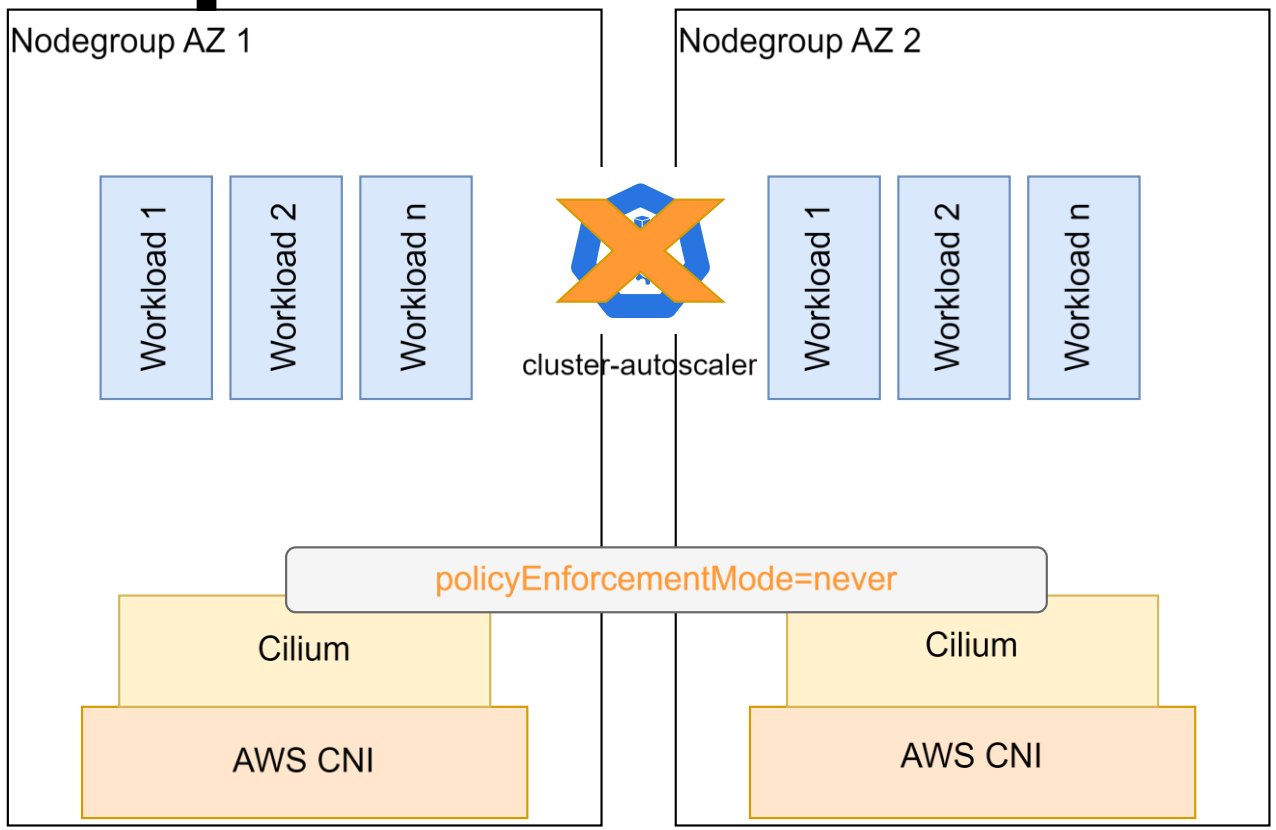
CNI-chaining problems

How do you migrate away from CNI-chaining without downtime?



kxcd.com

Step 0



Cilium runs on all current nodes

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: cni
          operator: In
          values:
          - both
```

AWS CNI runs on all nodes
No nodeAffinity

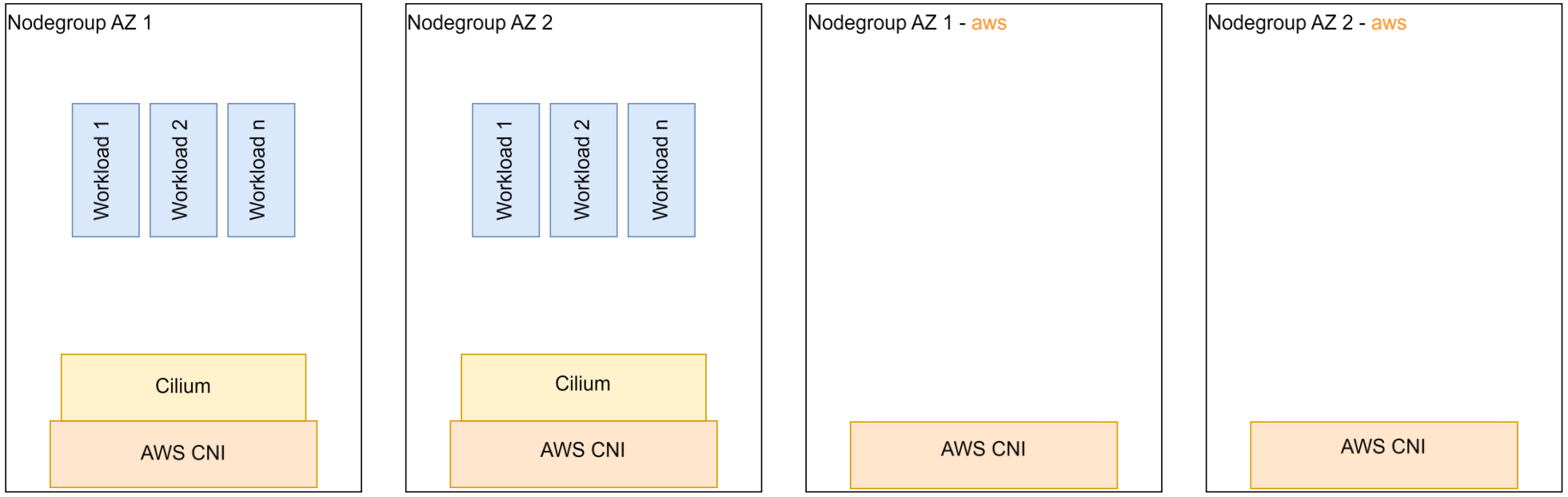
Custom taints:
- key: node.cilium.io/agent-not-ready
  value: "true"
  effect: NoSchedule

Custom Labels:
- cni=both

Custom taints:
- key: node.cilium.io/agent-not-ready
  value: "true"
  effect: NoSchedule

Custom Labels:
- cni=both

Step 1



Custom taints:
- key: node.cilium.io/agent-not-ready
  value: "true"
  effect: NoSchedule

Custom Labels:
  cni: both

Custom taints:
- key: node.cilium.io/agent-not-ready
  value: "true"
  effect: NoSchedule

Custom Labels:
  cni: both

Custom taints: []

Custom Labels:
  cni: aws

Custom taints: []

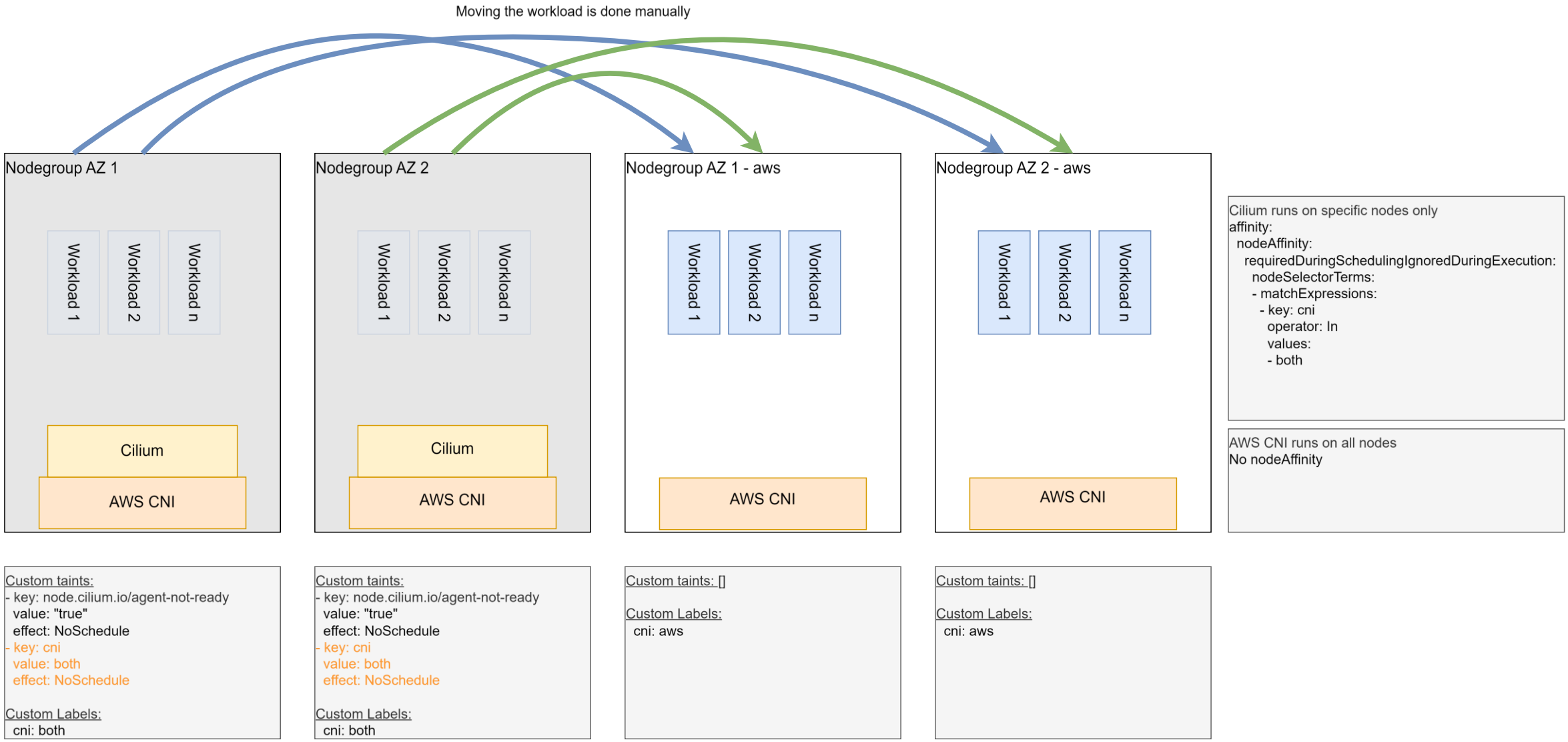
Custom Labels:
  cni: aws

Cilium runs on all current nodes

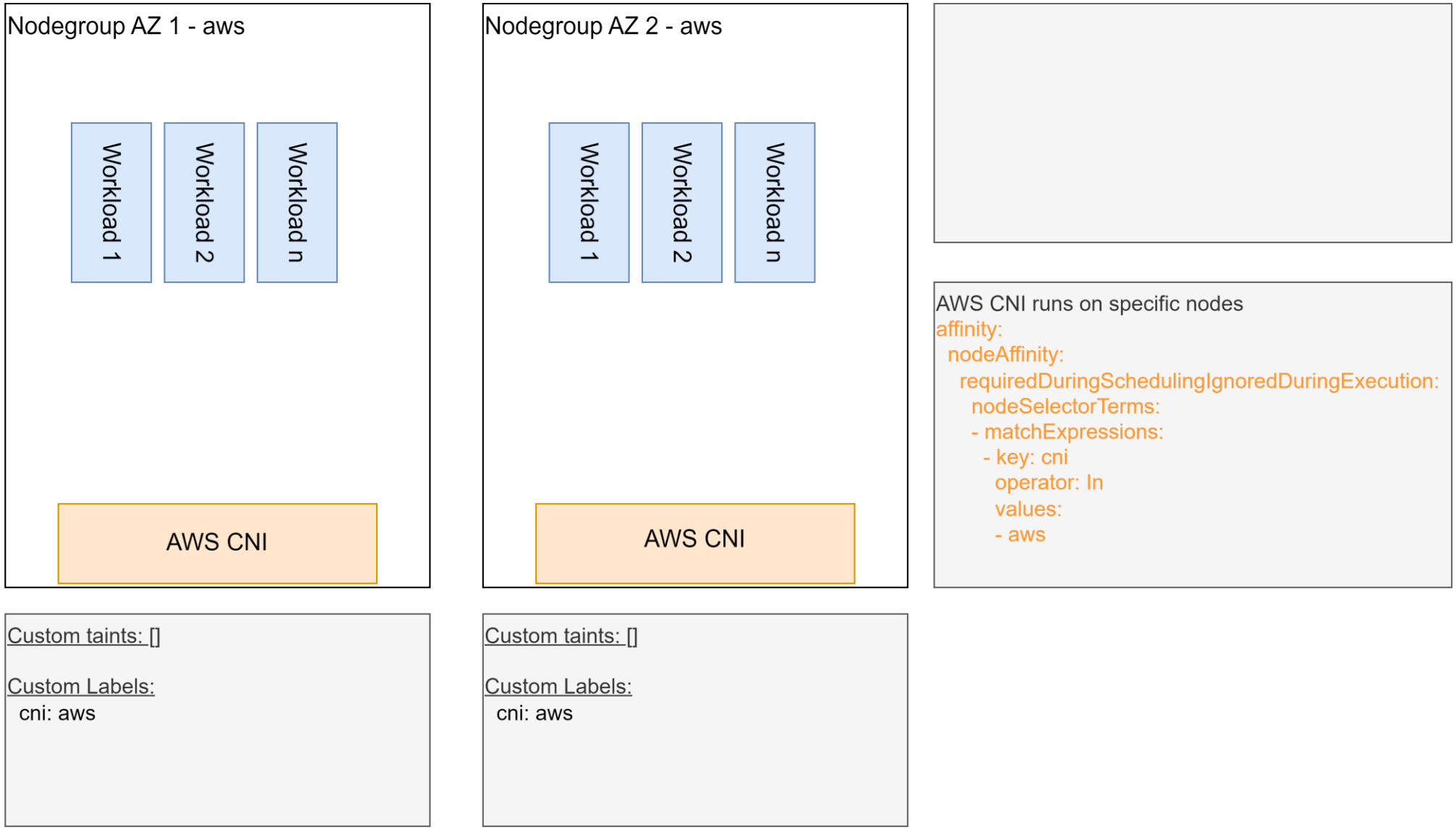
```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: cni
          operator: In
          values:
          - both
```

AWS CNI runs on all nodes
No nodeAffinity

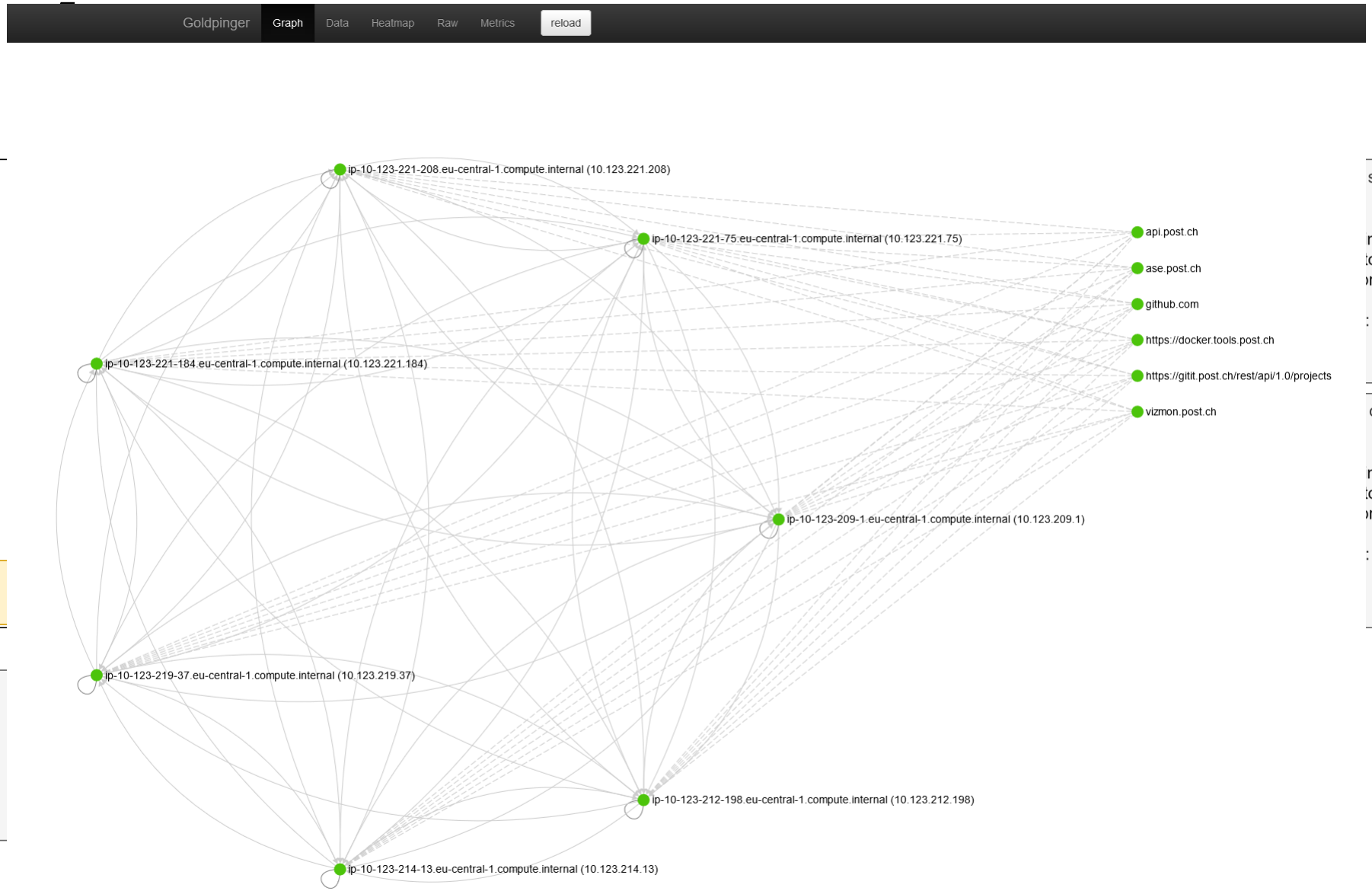
Step 2


Moving the workload is done manually
Nodegroup AZ 1
Workload 1
Workload 2
Workload n
Cilium
AWS CNI
Nodegroup AZ 2
Workload 1
Workload 2
Workload n
Cilium
AWS CNI
Nodegroup AZ 1 - aws
Workload 1
Workload 2
Workload n
AWS CNI
Nodegroup AZ 2 - aws
Workload 1
Workload 2
Workload n
AWS CNI
Cilium runs on specific nodes only
affinity:
nodeAffinity:
requiredDuringSchedulingIgnoredDuringExecution:
nodeSelectorTerms:
- matchExpressions:
- key: cni
operator: In
values:
- both
AWS CNI runs on all nodes
No nodeAffinity
Custom taints:
- key: node.cilium.io/agent-not-ready
value: "true"
effect: NoSchedule
- key: cni
value: both
effect: NoSchedule
Custom Labels:
cni: both
Custom taints:
- key: node.cilium.io/agent-not-ready
value: "true"
effect: NoSchedule
- key: cni
value: both
effect: NoSchedule
Custom Labels:
cni: both
Custom taints: []
Custom Labels:
cni: aws
Custom taints: []
Custom Labels:
cni: aws

Step 3


Nodegroup AZ 1 - aws
Workload 1
Workload 2
Workload n
AWS CNI
Nodegroup AZ 2 - aws
Workload 1
Workload 2
Workload n
AWS CNI
AWS CNI runs on specific nodes
affinity:
nodeAffinity:
requiredDuringSchedulingIgnoredDuringExecution:
nodeSelectorTerms:
- matchExpressions:
- key: cni
operator: In
values:
- aws
Custom taints: []
Custom Labels:
cni: aws
Custom taints: []
Custom Labels:
cni: aws

Step



Nodegroup AZ 1 - cilium

Cilium

Custom taints:
- key: cni
  value: "cilium"
  effect: NoExecute

Custom Labels:
  cni: cilium

specific nodes only

ngSchedulingIgnoredDuringExecution:
orTerms:
ressions:

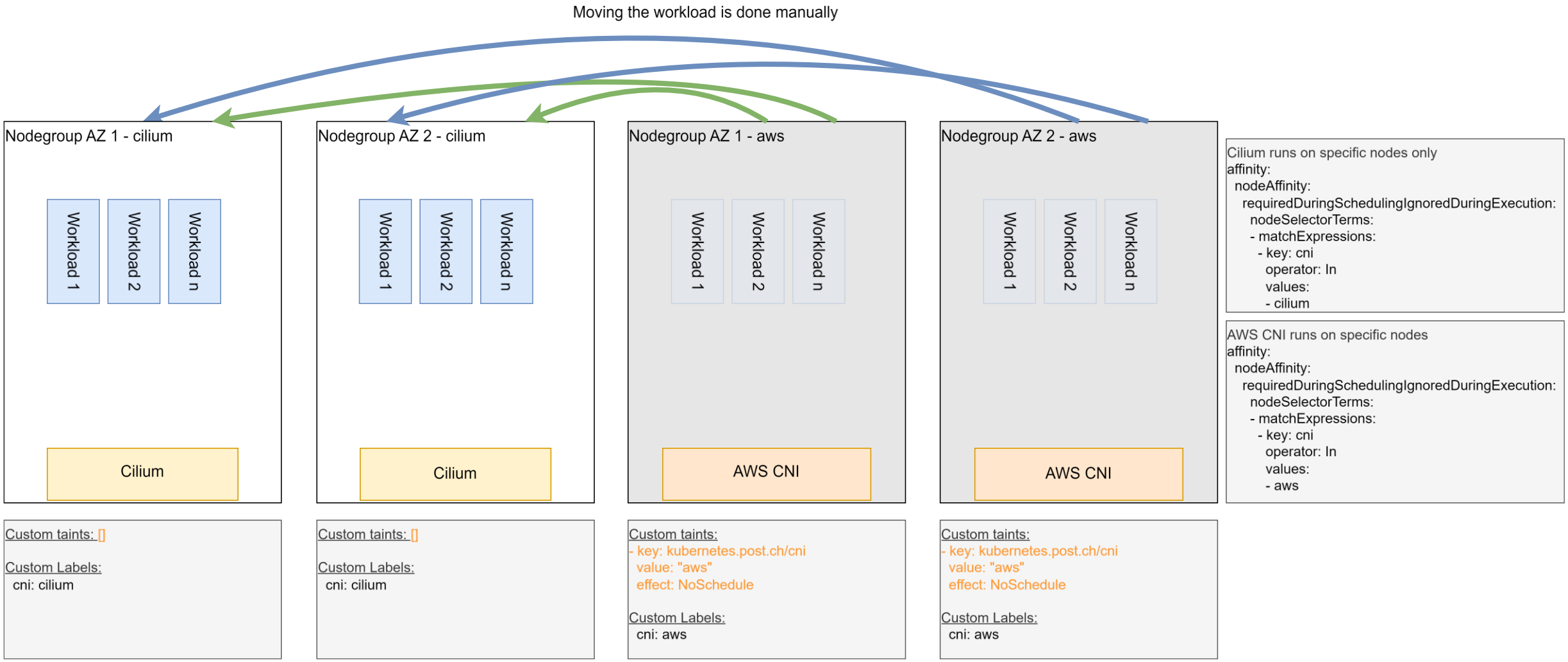
: In

on specific nodes

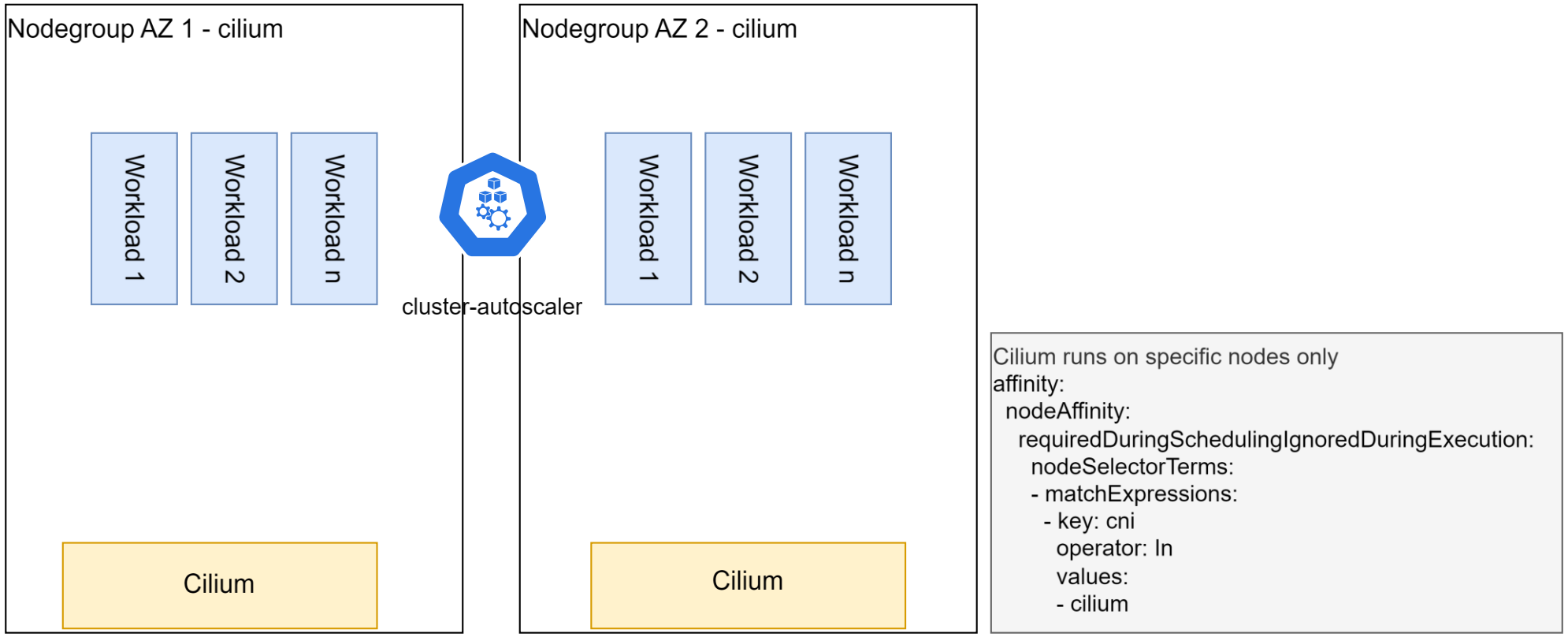
ngSchedulingIgnoredDuringExecution:
orTerms:
ressions:

: In

Step 5

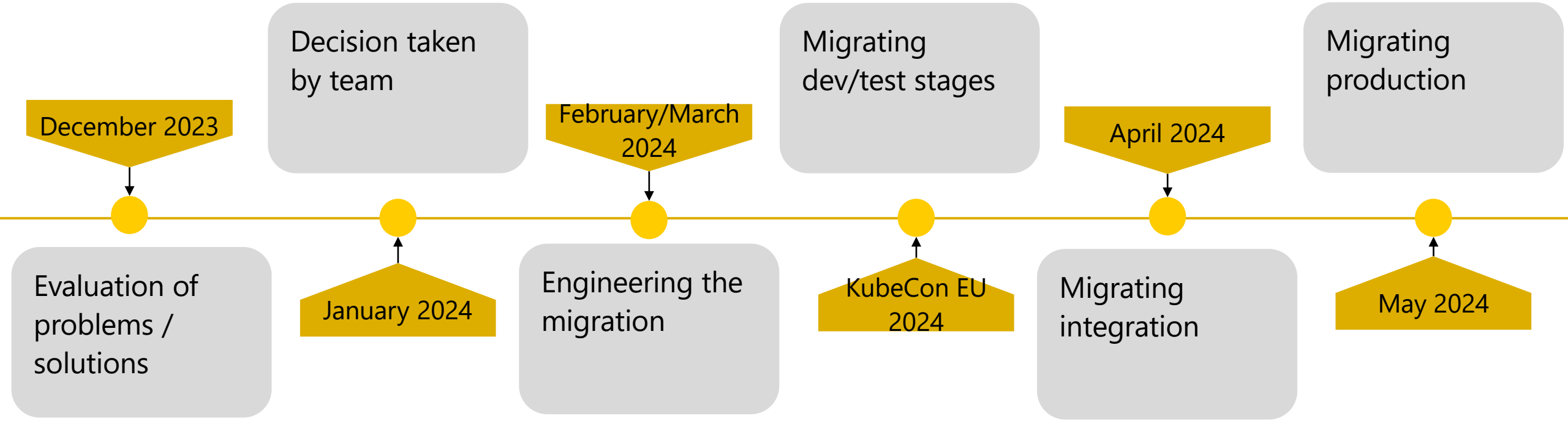

Moving the workload is done manually
Nodegroup AZ 1 - cilium
Workload 1
Workload 2
Workload n
Cilium
Nodegroup AZ 2 - cilium
Workload 1
Workload 2
Workload n
Cilium
Nodegroup AZ 1 - aws
Workload 1
Workload 2
Workload n
AWS CNI
Nodegroup AZ 2 - aws
Workload 1
Workload 2
Workload n
AWS CNI
Cilium runs on specific nodes only
affinity:
nodeAffinity:
requiredDuringSchedulingIgnoredDuringExecution:
nodeSelectorTerms:
- matchExpressions:
- key: cni
operator: In
values:
- cilium
AWS CNI runs on specific nodes
affinity:
nodeAffinity:
requiredDuringSchedulingIgnoredDuringExecution:
nodeSelectorTerms:
- matchExpressions:
- key: cni
operator: In
values:
- aws
Custom taints: []
Custom Labels:
cni: cilium
Custom taints: []
Custom Labels:
cni: cilium
Custom taints:
- key: kubernetes.post.ch/cni
value: "aws"
effect: NoSchedule
Custom Labels:
cni: aws
Custom taints:
- key: kubernetes.post.ch/cni
value: "aws"
effect: NoSchedule
Custom Labels:
cni: aws

Step 6


Nodegroup AZ 1 - cilium
Workload 1
Workload 2
Workload n
cluster-autoscaler
Nodegroup AZ 2 - cilium
Workload 1
Workload 2
Workload n
Cilium
Cilium
Cilium runs on specific nodes only
affinity:
nodeAffinity:
requiredDuringSchedulingIgnoredDuringExecution:
nodeSelectorTerms:
- matchExpressions:
- key: cni
operator: In
values:
- cilium
Custom taints: []
Custom Labels:
cni: cilium
Custom taints: []
Custom Labels:
cni: cilium

Migration Experience

Migration Roadmap



Learnings

Don't start with CNI-Chaining if you're sure you will end up with advanced features in production

Switching CNI is easier than changing something within a CNI

Aim for cloud-native apps that tolerate pod-churn

Thank you