

Platform Engineering Day NORTH AMERICA



Platform Engineering Day NORTH AMERICA

Data-Driven Platform Guardrails

What 16,500 Incidents Tell You to Automate First

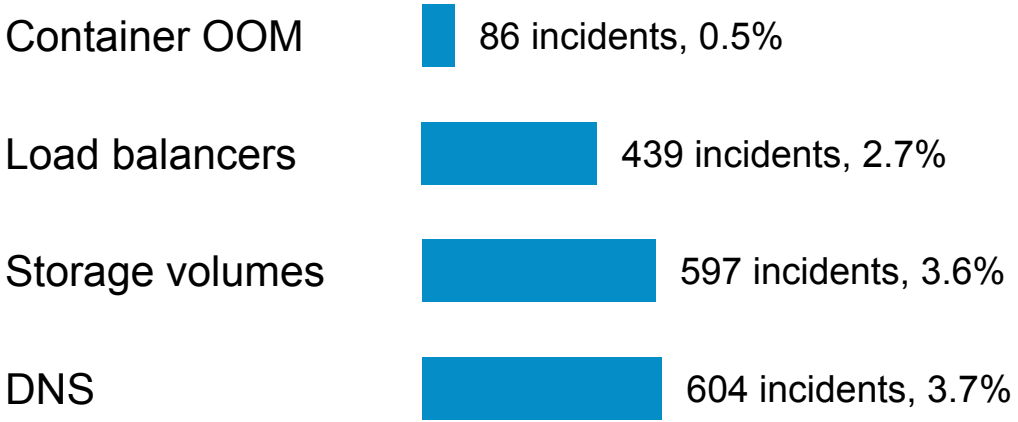
Henrique Santana,
Principal Cloud Engineer,
AWS

Henrique Galafassi Dalssaso,
Site Reliability Engineering Manager,
iFood

What is the #1 cause of Kubernetes incidents?

The usual suspects

How often they show up in 16,507 incidents



“It is always DNS".

In this data, DNS is 11th.

What exactly is this data

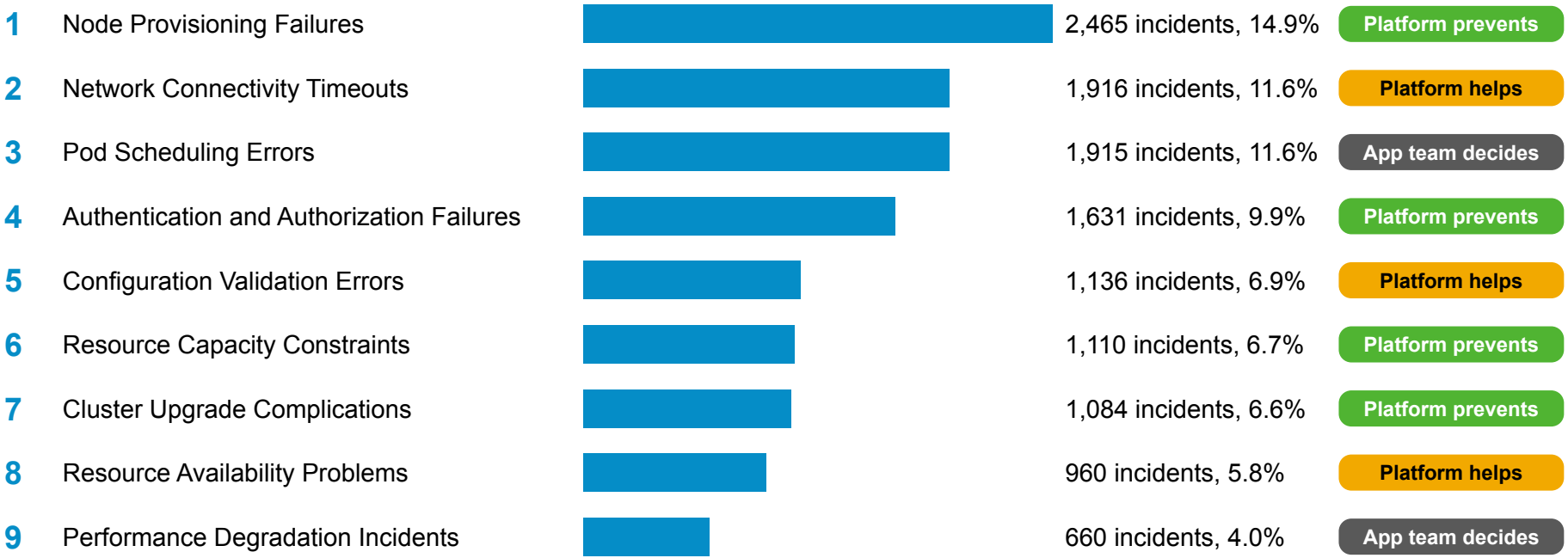
16,507 incidents across multiple Kubernetes clusters, **January 1 to May 13, 2026**

Classified by **root cause**. **AI-clustered** and **validated**

15 root-cause patterns, each with sub-patterns. 1.8% unclassified

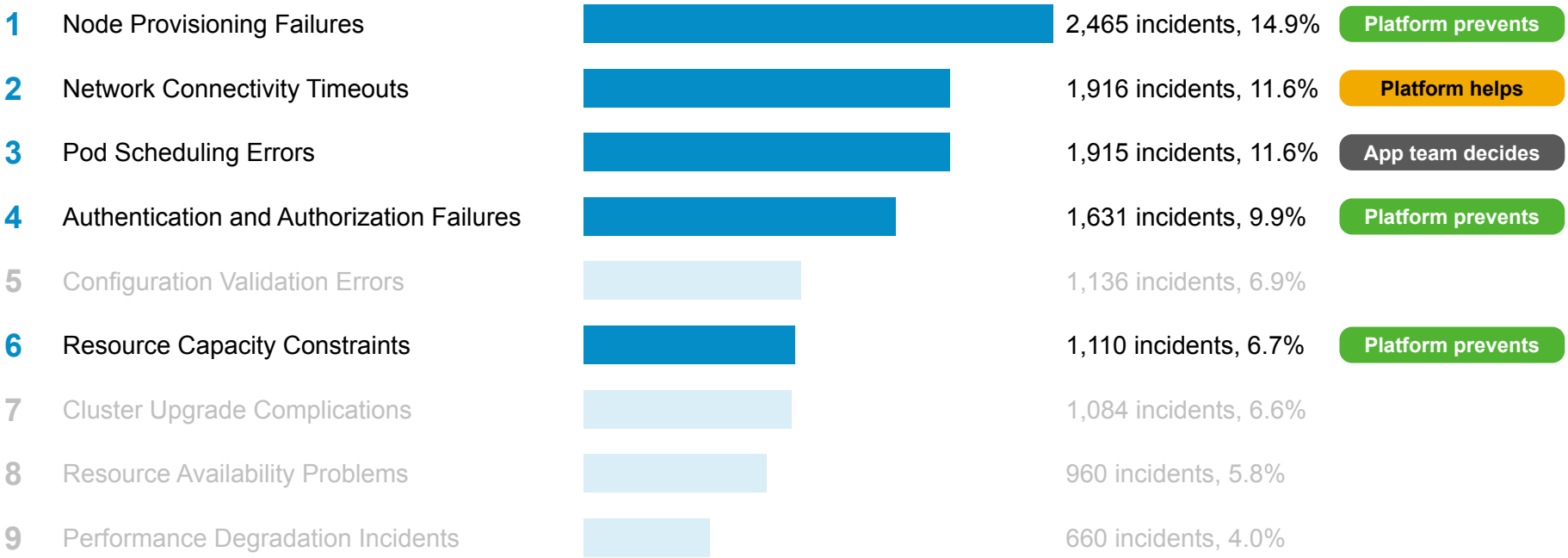
**Your ranking will differ*

9 patterns, 78% of incidents



Five patterns in depth

Platform Engineering Day NORTH AMERICA



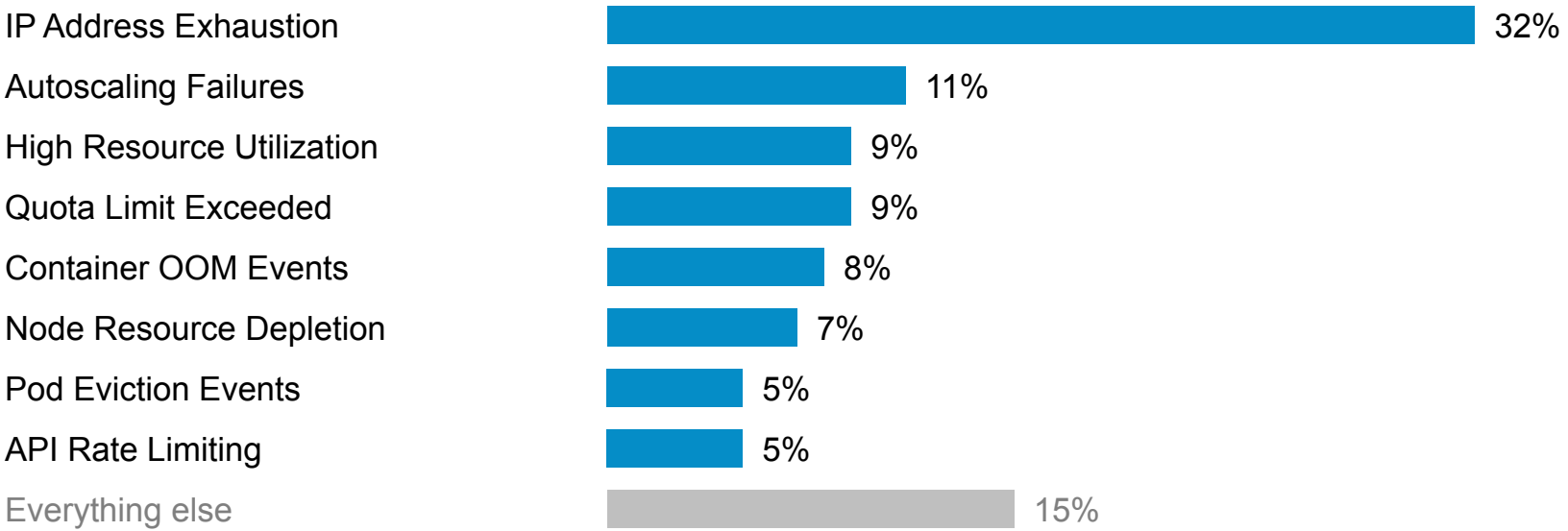
Smallest first. What it is, what causes it, how the platform can help.

Resource Capacity Constraints

1,110 incidents, 6.7%

The cluster ran out of something it needed.

Inside the pattern



No IP address available

One third of this pattern

Causes seen in this data

- Pod address ranges too small or fragmented
- CNI warm pools over-reserving addresses on every node
- Pods sharing the primary range with everything else

What the platform did

IP address exhaustion, 32%	**Cilium Multi-Pool IPAM**	Adds address ranges per node before the primary range runs out
Autoscaling and quota limits, 19%	**Karpenter NodePool limits**	Caps how much compute a NodePool can add, so scale-out stops at a known ceiling
High utilization and OOM, 16%	**Vertical Pod Autoscaler recommender**	Suggests right-sized CPU and memory before containers hit their limits

Platform prevents

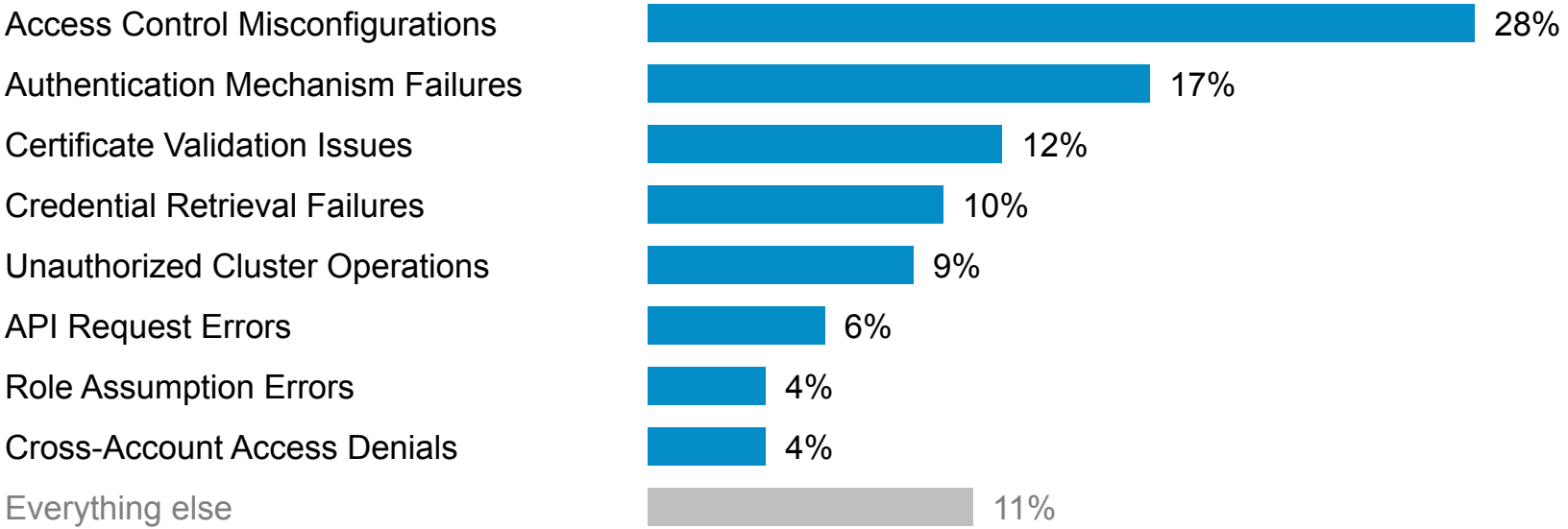
docs.cilium.io IPAM, karpenter.sh NodePools, kubernetes/autoscaler VPA.

Authentication and Authorization Failures

1,631 incidents, 9.9%

Someone or something could not prove who it was, or claimed too much.

Inside the pattern



Access denied

28% of this pattern

Causes seen in this data

- Missing or insufficient cloud identity permissions
- Missing cluster access mapping or RBAC binding
- Organisation-level policy or trust policy denying the call

What the platform did

Access control and unauthorized operations, 37%	**Kyverno on RoleBinding and ClusterRoleBinding**	In Enforce mode, denies bindings to system:masters and other risky groups
Authentication and credential retrieval, 27%	**ServiceAccount projected tokens**	Short-lived pod-bound tokens replace long-lived credentials teams forget to rotate
Certificate validation, 12%	**cert-manager**	Renews certificates automatically before they expire

These three cover 76% of the pattern.

Platform prevents

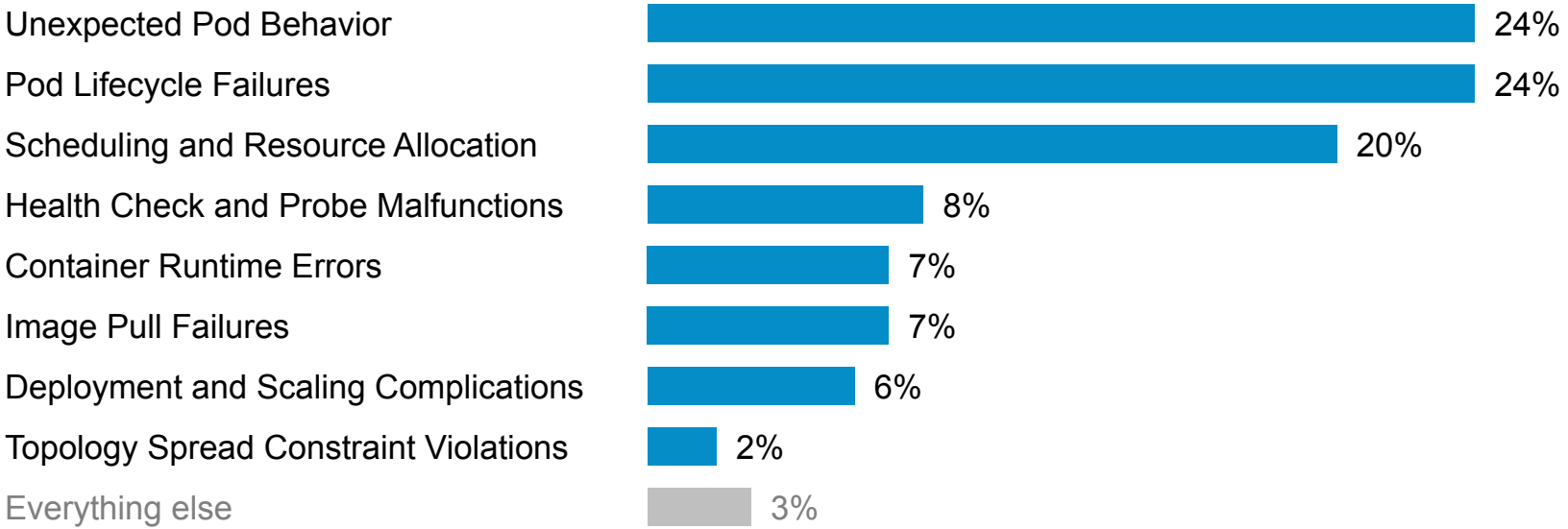
kyverno.io restrict-binding-system-groups, kubernetes.io service-accounts-admin, cert-manager.io.

Pod Scheduling Errors

1,915 incidents, 11.6%

The pod never ran, or ran and died.

Inside the pattern



CrashLoopBackOff

24% of this pattern

Causes seen in this data

- Memory or ephemeral storage exhausted
- Instance metadata or a private endpoint blocked
- Application startup or health endpoint failing

Causes here are long-tail. These three lead, others cluster just as close.

The other 24%, pods that never started, has no leading cause.

What the platform can still do

Crash loops and lifecycle failures, 49%	**Vertical Pod Autoscaler, updateMode Off**	Recommends right-sized CPU and memory. Startup bugs stay with the application owner
Scheduling and resource allocation, 20%	**LimitRange and Kyverno require-requests-limits**	LimitRange fills missing defaults. Kyverno in Enforce mode rejects the rest
Health check and probe malfunctions, 8%	**Backstage Tech Insights scorecards**	Flags services missing requests, limits or probes before they deploy

These three reach 76% of the pattern. The platform makes the gaps visible, the numbers stay with the application owner.

App team decides

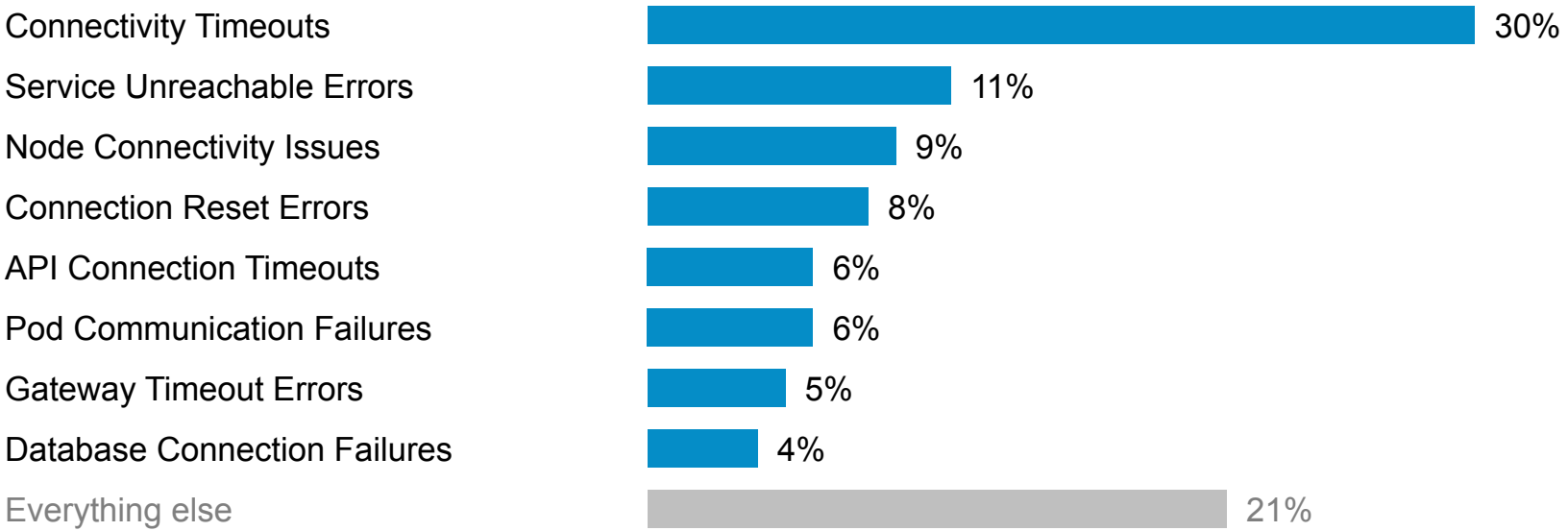
kubernetes/autoscaler VPA quickstart, kubernetes.io limit-range, kyverno.io require-pod-requests-limits, backstage community-plugins tech-insights.

Network Connectivity Timeouts

1,916 incidents, 11.6%

Packets did not arrive. Firewall rules, routing, missing private endpoints.

Inside the pattern



Connection timeout

30% of this pattern

Causes seen in this data

- Firewall rules blocking or missing
- Private service endpoints missing, NAT idle timeouts
- Routing misconfigured between networks

What the platform did

Connectivity timeouts, 30%	**Kubernetes ClusterNetworkPolicy, admin tier (Cilium 1.20)**	Cluster-wide baseline firewall rules that per-team policy cannot override
Service unreachable and pod communication, 17%	**Kyverno generate NetworkPolicy**	Creates a default-deny egress NetworkPolicy per namespace. Ingress needs its own rule
Connection resets and API timeouts, 14%	**Cilium Hubble**	Shows which policy or hop dropped the connection

These three cover 61% of the pattern.

Platform helps

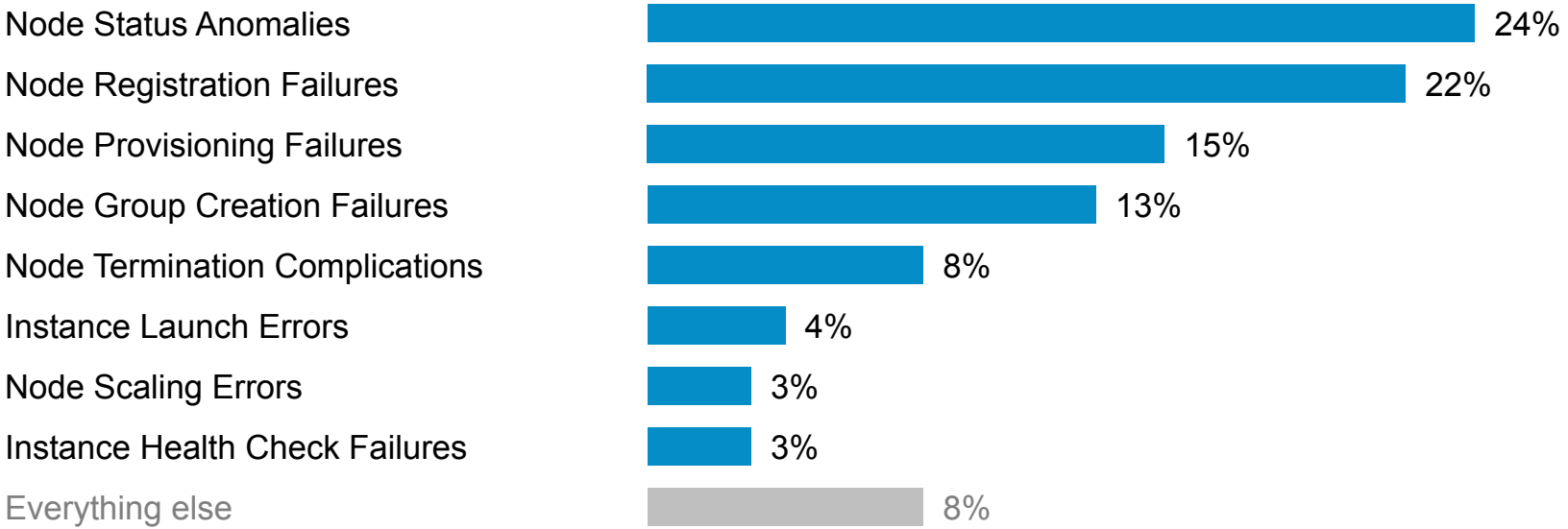
docs.cilium.io policy (v1.20), kyverno.io generate-networkpolicy-existing, docs.cilium.io hubble, docs.crossplane.io compositions.

Node Provisioning Failures

2,465 incidents, 14.9%

A node never joined, or joined and went NotReady.

Inside the pattern



Nodes NotReady

24% of this pattern

Causes seen in this data

- CPU or memory exhaustion starving the kubelet
- Kubelet cut off from the control plane, including missing private endpoints
- Disk or volume I/O throttling

The other 22%, nodes that never joined, leads with networking misconfiguration and bootstrap config errors.

What the platform did

Nodes going NotReady, 24%	**kubelet reserved resources (kube-reserved, system-reserved)**	Reserving CPU and memory for the kubelet stops workloads starving it
Registration failures and terminations, 30%	**MachineHealthCheck or Karpenter Node Auto Repair**	Replaces nodes that never register or go unhealthy past a timeout
Provisioning and launch failures, 19%	**Karpenter NodePool requirements**	Fewer instance-type restrictions give the provisioner more placement choices

These three cover 73% of the pattern.

Platform prevents

kubernetes.io reserve-compute-resources, cluster-api.sigs.k8s.io healthchecking, karpenter.sh disruption, karpenter.sh docs.

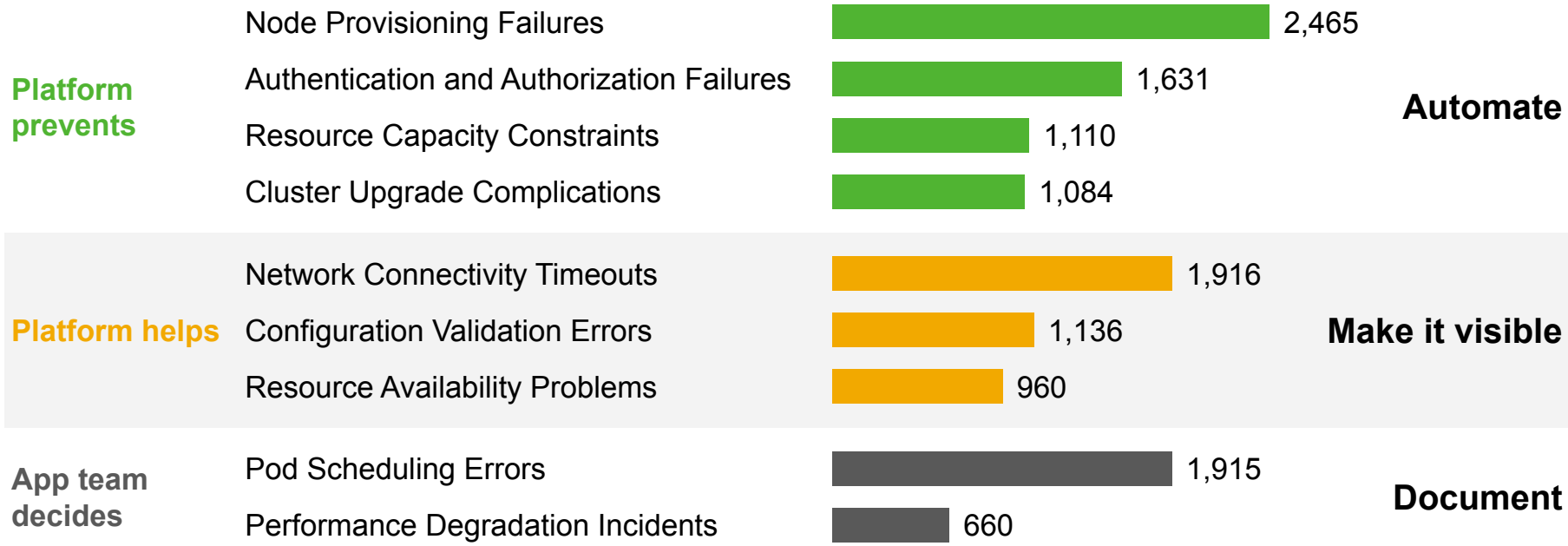
Four more, one line each

5th	**Configuration Validation Errors**	1,136	6.9%	**Platform helps**	Kyverno validate rules block manifests missing required fields before admission
7th	**Cluster Upgrade Complications**	1,084	6.6%	**Platform prevents**	Cluster API rolling upgrades orchestrate control plane and node version changes
8th	**Resource Availability Problems**	960	5.8%	**Platform helps**	Topology spread constraints across zones absorb a zone event. They do not prevent it
9th	**Performance Degradation Incidents**	660	4.0%	**App team decides**	API Priority and Fairness keeps the API server up under request floods

kyverno.io validate, cluster-api.sigs.k8s.io upgrading-clusters, kubernetes.io topology-spread-constraints, kubernetes.io flow-control.

Where to spend platform effort

Platform Engineering Day NORTH AMERICA



Bar length = incidents in this dataset

At least 38%

of the incidents in this dataset are structurally preventable at the platform layer.

Platform prevents 38.1%, platform helps 24.3%, app team decides 15.6%, long tail 22.0%.

Run this on your own data

1 **Classify**

Group incidents by what failed, not by what the user saw

2 **Rank**

By frequency. Who can fix it: platform, both, or team.

3 **Act**

Platform: automate. Both: make it visible. Team: document



Platform
Engineering Day
NORTH AMERICA

Thank you



Henrique Santana,
linkedin.com/in/henriquessantana

Henrique Galafassi Dalssaso,
linkedin.com/in/dalssaso

Resources



Platform Engineering Day NORTH AMERICA

https://kubernetes.io/docs/concepts/policy/resource-quotas/
https://kubernetes.io/docs/concepts/policy/limit-range/
https://kubernetes.io/docs/tasks/administer-cluster/reserve-compute-resources/
https://kubernetes.io/docs/reference/access-authn-authz/rbac/
https://kubernetes.io/docs/reference/access-authn-authz/service-accounts-admin/
https://kubernetes.io/docs/concepts/services-networking/network-policies/
https://kubernetes.io/docs/concepts/scheduling-eviction/topology-spread-constraints/
https://kubernetes.io/docs/concepts/cluster-administration/flow-control/
https://kubernetes.io/docs/concepts/workloads/autoscaling/vertical-pod-autoscale/
https://github.com/kubernetes/autoscaler/tree/master/vertical-pod-autoscaler
https://github.com/kubernetes/node-problem-detector/
https://karpenter.sh/docs/concepts/nodepools/
https://karpenter.sh/docs/concepts/disruption/
https://cluster-api.sigs.k8s.io/tasks/automated-machine-management/healthchecking
https://cluster-api.sigs.k8s.io/tasks/upgrading-clusters
https://kyverno.io/policies/other/restrict-binding-system-groups/restrict-binding-system-groups/
https://kyverno.io/policies/best-practices-cel/require-pod-requests-limits/require-pod-requests-limits/
https://kyverno.io/policies/other/generate-networkpolicy-existing/generate-networkpolicy-existing/
https://kyverno.io/docs/policy-types/cluster-policy/validate/
https://cert-manager.io/docs/usage/certificate/
https://docs.cilium.io/en/stable/network/concepts/ipam/
https://docs.cilium.io/en/stable/observability/hubble/
https://docs.cilium.io/en/v1.20/network/kubernetes/policy/
https://docs.crossplane.io/latest/composition/compositions/
https://github.com/backstage/community-plugins/blob/main/workspaces/tech-insights/plugins/tech-insights/README.md