

OneK8s

One Kubernetes platform, one application contract for every cloud - built on Azure (AKS) first.

Requirements, version 4.5, built 2026-10-26 from commit [9f1f777](#).

The current version is published at <https://onek8s.github.io>; the changes between versions are in [8.1 Release notes](#).

Contents

0. Requirements

[0.2 Requirements](#)

[0.3 Acceptance criteria](#)

0.2 Requirements

The requirements of the platform, owned by humans ([how we work](#)): a change to a requirement needs a human's approval. The acceptance column names the [acceptance criteria](#) that prove each requirement; the requirement and isolation tests are tagged with the requirement number they cover. How the design meets each requirement is design [section 1.2](#).

#	Requirement	Acceptance
R1	Internal and external workloads isolated – at subscription and cluster level (security, 2026-10-14)	Isolation tests : one exposure per cluster and subscription
R2	Workloads of different countries isolated – node pool and network are enough	Isolation tests : every flow between zones blocked
R3	Isolation = Key Vault + network + node pool per type	Isolation tests : another zone's pods, nodes and Key Vault blocked; every zone complete
R4	Only connectivity mandatory for Kubernetes allowed between zones	Isolation tests : every flow between zones blocked and logged by the Firewall
R5	Countries can be added online	Isolation tests : every zone complete; adding a country in upgrades and changes
R6	Azure CNI powered by Cilium in every cluster	Design review
R7	Control plane in a dedicated network	Isolation tests : the API server only from the management VNet; local admin and SSH refused
R8	Cost-optimised compute: applications choose amd64 or arm64 and on-demand or spot nodes, the platform supports all of them	Performance : scale-out with arm64 and spot, spot eviction
R9	Workload identities wherever possible, secrets only when needed	Isolation tests : no local auth found by Resource Graph
R10	Key Vault multi-tenancy with Azure RBAC + ABAC	Isolation tests : another application's secret refused by ABAC
R11	Applications never read Key Vault directly; External Secrets Operator is mandatory for them	Isolation tests : an application that reads Key Vault itself is refused
R12	Certificates distributed through Key Vault, shared by all clusters	Isolation tests : the certificate from the vault to the listener; renewal in upgrades and changes
R13	Applications isolated in namespaces + network policies	Isolation tests : namespace groups and roles

#	Requirement	Acceptance
R14	Every workload's owner and cost are known	Forbidden manifests in the isolation tests ; cost in operations
R15	One policy engine	Isolation tests : every forbidden manifest denied by admission
R16	Applications and how they are deployed do not change between clouds (portability of the application contract)	Isolation tests : application repositories scanned for cloud-specific objects
R17	Two cluster types: stateless and stateful	Design review
R18	Stateful cluster is the last option	Design review
R19	Stateful cluster exists at most once per exposure (R1), in the backend network with the Azure PaaS services, no Internet egress and no public endpoints ; the only outbound destinations outside Private Link are allow-listed Microsoft control-plane endpoints that have no Private Link (Entra ID for workload identity, the Azure Policy add-on)	Isolation tests : no Internet destination from the stateful cluster but the documented exceptions
R20	Stateful cluster spread over availability zones and running AKS LTS	High availability : AZ outage (stateful cluster, if built)
R21	Stateful cluster: VNet-integrated CNI, no Gateway API implementation (and no Ingress), applications handle TLS	Design review
R22	Stateless clusters: pod subnet CNI, no service mesh / mTLS / application TLS, TLS-only Gateway API (no Kubernetes Ingress); from outside its namespace an application is reached only as HTTPS on 443 through the zone's gateway, inside its namespace it uses any protocol	CIS benchmark : TLS on every gateway listener; forbidden manifests in the isolation tests
R23	Stateless: separate cluster per availability zone (at least two), ≥ 2 copies of every serving application, scaled on load; workloads that serve no request (jobs, scheduled jobs, queue consumers) are allowed with the copies their work needs, and work that must not repeat runs once per environment	Isolation tests : the cell's settings; performance and high availability
R24	Every long-running workload has health probes, and every application published through a gateway a health endpoint the platform probes; a finite job reports its result by its exit status	Forbidden manifests in the isolation tests ; failover in high availability
R25	dev, acc and prd environments kept in sync	Isolation tests : every cell in sync with Git and its IaC; rebuild
R26	Fully automated zero-downtime upgrades of applications and clusters	Zero-downtime upgrades and changes
R27	Update policy: stateful cluster only when absolutely necessary, stateless clusters at two speeds	Zero-downtime upgrades and changes : minor and node image upgrades
R28	Data at rest encrypted with customer-managed keys: own key for each control plane and each node pool, stored in the platform Key Vault	CIS benchmark : KMS; design review

#	Requirement	Acceptance
R29	Traffic exiting the cluster is allowed only to named destinations (FQDNs)	Isolation tests : egress to an unnamed FQDN or an IP address denied
R30	Day 2 operations driven by AI agents, also in prd, but only through GitOps / IaC pull requests within human-defined guardrails	Upgrades and changes : an agent change through every risk tier and the kill switch
R31	Observability, including tracing, with Grafana Cloud; anyone allowed to connect to it may see all data	Operations : telemetry and no secrets in it
R32	Protect against losing the region with immutable backups in another region; back up only persistent data, not whole clusters	Rebuild, backup and recovery
R33	The platform's rules are non-negotiable and the same for every application: its patch timetable binds every application (base images are rebuilt and promoted on the platform's schedule, in acc and prd on automated tests alone), automated tests are the only gate of a release, and an application that cannot meet the rules is not onboarded or carries an exception with an expiry date	Zero-downtime upgrades and changes : a base image patch to prd without a human; Isolation tests : an image past its deadline refused
R34	Every team can test a change before it reaches dev: a preview environment per pull request and a longer-running sandbox per team, inside dev and under the same guardrails as dev	Isolation tests : a preview reaches exactly what its application reaches and is removed with its pull request

0.3 Acceptance criteria

The acceptance criteria prove the [requirements](#): the requirement and isolation tests name the requirement each covers, and the platform is accepted only when every test below passes. Like the requirements, they are owned by humans ([how we work](#)). Where and when the tests run - the build tenant, the phases, the security reviews and the connection readiness gate - is the implementation project's plan, outside the published design.

Acceptance tests in dev

A **reference application** is deployed into every isolation zone of every cell before acceptance starts: an HTTP API with a frontend in the stateless cells, a PaaS database through workload identity, a Key Vault secret exception, a Firewall rule of its own and – if the stateful cluster is built – a StatefulSet with three replicas. Load is generated from the Internet through Traffic Manager for `ext-*` zones and from the client VNet directly to the home cell for `int-*` zones, and the generators' own error rate and latency are the measurement in every test below. Targets marked *TBD* are agreed with the application owners before phase 3 starts.

Switch policies to `deny` first. Before any acceptance test, every Azure Policy assignment – corporate initiatives and the platform's own definitions – is switched to its production effect in the build tenant. A test that only passes in audit mode has not passed.

Requirement and isolation tests

Automated negative tests, kept from phase 2, run as a pipeline stage after every change of a cell and daily. Each check names the requirement it covers. A forbidden manifest refused by a rule that is still in `audit` is reported as *audit only* rather than as a pass; it becomes a failure once that rule must deny:

Test	Expected result
From a pod in every zone, connect to every other zone's pods, ILB, nodes, Key Vault and private endpoints, in the same cell, the other cells and the stateful cluster; and to another zone's published name that the pod's rule file does not list	Blocked: the Firewall logs the drop of every flow between zones, the NSGs and Cilium stand behind it (section 2.4); the published name is open only with an application rule in the caller's rule file, and then through the Firewall's proxy, never directly
Application A's <code>ExternalSecret</code> references application B's secret in the same zone Key Vault	Sync fails with <code>403</code> from Key Vault ABAC (section 2.5)
An application reads Key Vault itself: SDK call with its own identity, a pod running as <code>secrets-reader</code> , a <code>SecretProviderClass</code> or CSI volume, a hand-made <code>Secret</code> , a vault name in its egress policy	Dropped by Cilium / <code>403</code> (no role) / denied by admission; an Azure Resource Graph query finds no Key Vault data-plane role on any application identity

Test	Expected result
Egress to an FQDN not in the zone's Firewall rules, and to an IP address; an application policy that tries to open either	Denied and logged by the hub Firewall (section 5.1); the policy refused by admission (section 2.7)
From a pod of application A, connect to application B of the same zone on its <code>Service</code> , its pod addresses and any port, and to its published name; between A's own pods, any protocol and port (R22)	B's <code>Service</code> and pods refused; the published name answers only with a rule in A's rule file, over HTTPS on 443 through the zone's gateway; everything between A's own pods passes
From a pod in every zone, request a token from the instance metadata service (<code>169.254.169.254</code>)	Dropped by Cilium
A member of a namespace's <code>view</code> , <code>edit</code> and <code>admin</code> group (impersonated, and signed in as a test application in the group) reads, writes and binds roles in that namespace, and does anything in another namespace or cluster-wide	Exactly the role's rights in its namespace (<code>view</code> no <code>Secret</code> s, only <code>admin</code> role bindings), refused everywhere else; the namespace's <code>RoleBinding</code> s match the IaC; in the zone vault nobody lists, <code>view</code> has nothing, <code>admin</code> reads and writes only the namespace's prefix by name, <code>edit</code> writes and updates it but reads no value and deletes nothing, and neither creates anything outside it (section 2.5)
A platform viewer lists the clusters' namespaces and the vaults' secrets, reads a pod, a <code>Secret</code> or a secret value, writes	Lists both (names only); everything else refused; its role is assignable at the environment's resource group only (section 2.6) (section 2.6)
Sign in to every cluster with the local admin credential (<code>az aks get-credentials --admin</code>) and SSH to a node	Both refused: local accounts and SSH are disabled (section 2.2); <code>--admin</code> is tried, SSH is checked as the cluster setting
Promote an image that is not signed by the platform's CI; push an image to the acc registry with any identity other than the promotion pipeline	Promotion fails on the signature check; the push is refused (no role); the registry's admin and anonymous access are off and the kubelet has only <code>AcrPull</code> . An unsigned image deployed in dev shows up as non-compliant in Image Integrity (section 2.7); every image in the registry and every image an application runs verifies against the platform's certificate
Apply a <code>Job</code> , a <code>CronJob</code> with <code>concurrencyPolicy: Forbid</code> and a queue consumer <code>Deployment</code> with one replica and no route, then a serving <code>Deployment</code> with one replica (R23, R24)	The three non-serving workloads are admitted (the consumer with its probes and graceful shutdown); the serving one is refused
Apply each forbidden manifest of the policy table (privileged pod, image not from ACR, <code>toCidr</code> , <code>PersistentVolumeClaim</code> in a stateless cluster, ...)	Denied by admission
Reach the API server from the client VNet, the hub and a spoke	Only from the management VNet (section 2.2)
Azure Resource Graph: public IPs, public endpoints of PaaS / Key Vault / ACR, local auth enabled	None except the designed ones (the external zones' gateway load balancers) and the recorded exceptions in a list kept next to the test (the Basic registry of dev and acc; AKS-managed disks and Private Link scopes)

Test	Expected result
Stateful cluster: any Internet destination	Blocked, apart from the documented Entra ID and Azure Policy exceptions
A preview namespace of the reference application (R34): from it, reach the application's own data store and its allowed destinations, another namespace, another zone, Key Vault; write a secret under another prefix; close the pull request	Reaches only what the application's namespace reaches, with the same identity and rule file; everything else denied like for the application; the namespace, its route and DNS name are gone within the interval after the close; a team's sandbox namespace meets every rule of the policy table
Deploy the reference application from an image whose base is past its patch deadline (R33)	Refused at admission in every cell; the running pods are untouched
Scan the application repositories for cloud-specific API groups, annotations, resource names and SDKs (R16)	None: only open-standard objects and the platform's names (<code>key-vault</code> store, zone gateways)

The tests also check what the table does not list: one exposure per cluster and subscription (R1), each namespace's groups and roles (R13), the cell's settings (R23), every cell in sync with Git and its IaC (R25), the certificate from the vault to the served listener (R12) and every zone complete (R5); its summary table names the requirement of each step.

CIS benchmark

The [CIS Azure Kubernetes Service \(AKS\) Benchmark](#) is the reference hardening checklist ([section 2.7](#)). Like the isolation tests, it is an automated stage that runs after every change of a cell and weekly, and it stays in the pipelines for the platform's life. It has three sources, merged into one result per CIS recommendation and a score (the share of automated recommendations that pass):

Benchmark section	Source	What is checked
3 Worker nodes	kube-bench (<code>aks-1.8</code>), one read-only Job per node	Kubelet configuration files, their permissions and owners; kubelet flags (anonymous auth, authorization mode, client CA, read-only port, certificate rotation, ...)
4 Policies	Kubescape (<code>cis-aks-t1.8.0</code>) against the API server, read-only, nothing installed	RBAC (cluster-admin, secrets, wildcards, pod creation, token creation, webhooks, ...), default service accounts and token mounting, Pod Security per namespace, a network policy in every namespace, secrets as files, the <code>default</code> namespace
2 Logging, 5 Managed services	The cluster's Azure settings	Audit logs to Log Analytics, image scanning by the company's security tool, the registry (no admin or anonymous access, kubelet <code>AcrPull</code> only), Azure RBAC with local accounts off, KMS, the API server's exposure, private nodes, network policy, TLS on every gateway listener, Entra ID

A failure is either fixed or **accepted with its reason** in a list kept next to the test, optionally for named objects only; any other failure, including a new object that fails an accepted recommendation, fails the stage. The list holds three

kinds of entries: dev's own exceptions, which acc and prd do not have (the Basic registry); what AKS manages and cannot be changed (its control-plane users, the add-ons' RBAC and namespaces); and design decisions with their section (the API server's public FQDN resolving to a private address instead of an AKS-managed private DNS zone, 5.4.2). The score and the result per recommendation are kept for every run, and a drop raises an alert to the security agent ([section 6.1](#)).

Target. In dev no unaccepted failure and no drop; the baseline is recorded in phase 2. In acc and prd dev's exceptions are gone, so those recommendations pass there; what remains accepted is only what AKS manages and the documented design decisions. Follow-ups that would raise the score further, each a change through phase 2:

- Network policies for the AKS add-on namespaces (`aks-istio-system` , `gatekeeper-system`) if AKS supports them there (4.4.2), and restricted Pod Security for `flux-system` once the Flux extension's pods meet it (4.2.8).
- The gateways' namespaces on restricted Pod Security once App Routing lets its gateway pods set a seccomp profile (4.2.8, [section 2.7](#)).

Performance and capacity

Test	How	Pass criterion
Peak load per cell	One cell takes 100 % of the expected prd peak (the other cells drained), for 1 hour	p95 / p99 latency and error rate within targets (<i>TBD</i>); no throttling of Firewall, the gateways' load balancers, Key Vault or Entra ID
Scale-out from idle	Load ramps from 10 % to 100 % of peak in 5 minutes	HPA / KEDA and the cluster autoscaler add capacity before latency breaches its target; time to new nodes measured per VM family, arm64 and spot included
Spot eviction under load	Simulated eviction of all spot nodes of a zone	Pods move to on-demand without errors beyond the budget; no SLO breach
In-place right-sizing under load	Reference application with the default VPA (<code>InPlaceOrRecreate</code>) and requests set 4× too high, then 4× too low for memory; a resize that cannot fit its node (<code>Deferred</code>) and one that never fits (<code>Infeasible</code>); a multi-container pod with a pod-level budget resized in place; a second application with <code>platform/rightsizing: off</code>	VPA converges within its bounds without errors beyond the budget; the opted-out application is never resized; CPU resizes cause no container restarts; deferred resizes end in an eviction and a new node, infeasible ones raise the right-sizing alert; HPA and VPA do not oscillate; the right-sizing agent's pull requests pass the gate as tier low
Soak	72 hours at 60 % of peak	No growth of memory, connections, SNAT port use or latency
Network dataplane	Pod-to-pod, pod-to-ILB and pod-to-PaaS throughput and latency (<code>netperf</code> / <code>iperf3</code>) with VNet encryption; per node under the peak load the share of CPU in softirq, the packet rate and the gateway-to-pod p99	Recorded as baseline; no regression later without an explanation. The eBPF host routing thresholds of section 5.1 are checked against it, and the same numbers are dashboards in production

Test	How	Pass criterion
DNS	CoreDNS query rate and latency at peak	No <code>SERVFAIL</code> / timeouts; CoreDNS sizing set
Control plane	Flux reconciling all zones and many application namespaces; Azure Policy admission latency; API server request latency	Reconcile time and admission latency within targets; no API throttling
Stateful cluster (if built)	Disk IOPS / latency on ZRS storage for the reference StatefulSet	Meets the workload's requirement

High availability and resilience

Fault injection with [Azure Chaos Studio](#) and Kubernetes chaos experiments, every one under load. Each experiment must also raise the expected alert. Chaos Studio's AKS faults run through Chaos Mesh, which installs a privileged daemon on every node. In dev of the build tenant it is installed only for the duration of the tests, in a platform namespace with a time-limited exemption ([section 2.7](#)); whether its network faults work with Cilium's policies and a private cluster is checked first. VM- and zone-level faults (stop VMSS instances, zone down) need no in-cluster agent.

Fault	Pass criterion
Loss of a whole cell (stop all nodes of <code>sl-az1</code> , then <code>sl-az2</code>)	Traffic Manager moves external traffic to the remaining cells within the target (<i>TBD</i> , probe interval + TTL); the alert-driven failover migrates the stopped cell's internal applications to another cell within the internal recovery time target (<i>TBD</i>); errors within budget; the surviving cell carries 100 %
AZ outage (Chaos Studio zone-down on all VMSS of one AZ, stateless and stateful)	Same as above for stateless; the stateful cluster keeps quorum and serves from two AZs
Node failures: kill system pool node, application node, the node of a gateway replica	No user-visible errors beyond budget; PDBs respected
Platform components: restart Cilium agents, CoreDNS, the cell's gateway implementation (istiod or Traefik) and the gateway pods, Flux, the Azure Policy add-on	Cilium's policies stay enforced during its restarts; traffic continues; admission fails closed as designed
Dependency loss: PaaS failover (zone-redundant database), Key Vault unavailable for External Secrets Operator, ESO controller of a zone stopped, Entra ID token endpoint slow	Applications keep running and new pods start with cached credentials and the Secrets already in the cluster; failed syncs alert; recovery without manual steps
Traffic layer and placement: disable a cell's Traffic Manager endpoint; migrate internal applications back and forth under load; stop ExternalDNS in one cell during a migration	Failover within target; cell affinity re-established; planned migrations without failed requests; DNS records always point to a cell that serves the application

Fault	Pass criterion
Firewall rule mistake (block a platform FQDN in one cell)	Detected by monitoring, contained to one cell

Shared tier failure domains

The shared tier of [section 3.1](#) is not split into cells, so each of its parts is a failure domain of its own; the fault table above injects each of these:

Shared component	Redundancy	If it fails	Effect on running traffic	Recovery
Azure Traffic Manager	Global DNS service, outside the region	DNS answers stop or go stale	Clients keep cached answers for the TTL; new clients cannot resolve external names	Microsoft; the per-cell host names still resolve through public DNS as a manual bypass
Hub Azure Firewall (one per exposure)	Zone-redundant; the other exposure's Firewall is untouched	Egress from every cell of that exposure stops – and, for the internal Firewall, the corporate path to the internal zones	External zones keep serving (their ingress does not pass the Firewall); workload identity token refresh and PaaS calls fail once cached tokens and connections expire	Microsoft; nothing in the cells changes
Hub Private DNS and resolvers	Zone-redundant service	Internal names and PaaS private endpoint names stop resolving for new connections	Cells keep established connections	Microsoft; IaC recreates the zones in a region recovery
Microsoft Entra ID	Global, multi-region	No new workload identity tokens, no human sign-in	Pods keep running on tokens already issued (up to their lifetime); ESO keeps the <code>Secret s</code> it synced; new pods of workload-identity applications start but cannot call Azure until Entra ID returns	Microsoft; this is why secrets are delivered as <code>Secret s</code> and not fetched at start
Azure Policy endpoints	Public endpoints, global	The add-on cannot fetch new assignments; admission keeps the constraints it has (fails closed on webhook errors as configured)	None for running pods; rollouts continue with the last synced policies, bootstrap of a new cell waits	Microsoft

Shared component	Redundancy	If it fails	Effect on running traffic	Recovery
Azure Key Vault (zone or platform)	Zone-redundant, geo-replicated within the region pair	ESO syncs fail and alert; KMS cannot decrypt new etcd reads if the KMS key is unreachable for long	Existing secrets stay usable; pods keep running; a long KMS outage degrades the API server of every cluster	Microsoft; Key Vault is the single most important shared dependency, which is why nothing fetches from it at request time
ACR	prd: zone-redundant Premium, geo-replicated to the paired region; dev and acc: Basic	No image pulls	Running pods unaffected (images cached on nodes); scale-out onto fresh nodes and rollouts stall	Microsoft; releases are halted by the operations agent
PaaS data services	Zone-redundant per service, one per zone and application	Only that application's data store	Only that application, in every cell at once (data is shared)	PaaS failover; point-in-time or geo-restore (section 6.3)
Stateful cluster	Zone-redundant control plane, node pools per AZ	Only the applications with a stateful exception, and their consumers	Consumers in every cell at once	In-place recovery, PDB-guarded; restore from the vault tier
Grafana Cloud and its Private Link	Outside the region	No new telemetry, no SLO gates	None for traffic; releases and upgrades stop at their gates (no data means no promotion); Alloy buffers on the nodes	Grafana Labs; the watchdog alert fires from outside the region
GitHub	Outside Azure	No merges, no Flux source	None for traffic; Flux keeps the last applied state; safe runbook actions still run from the runbooks repository mirror in the management VNet	GitHub; a region recovery does not depend on the lost region, but it does depend on GitHub

Zero-downtime upgrades and changes

The procedures of [section 3.4](#) and [section 3.5](#) are run end to end under load, at least once each; the pass criterion for all is **no errors beyond the budget at the load generators**.

- Application release cell by cell, including a database schema change and a rollback.
- A cell drained while a job of the reference application runs and a scheduled job of it falls due: the job finishes or is restarted in the target cell, and the scheduled run happens exactly once in the environment (R23).
- Kubernetes minor upgrade of `sl-az2` and of `sl-az1` (drain → upgrade → return), and a node image upgrade.
- A cell **deleted and rebuilt** from IaC and Flux while the others carry the traffic.

- Stateful cluster (if built): node image and patch upgrade one AZ at a time.
- Certificate renewal and rotation in the zone's gateway ([section 4.1](#)).
- Azure Policy definition change rolled out audit → deny.
- **Adding a country online** (two new zones, [R5](#)).
- An AI agent change through every risk tier, and the kill switch ([section 6.1](#)).
- A VPA bound change by the right-sizing agent rolled through the cells while the VPA resizes the reference application in place.
- **A base image patch to prd without a human** (R33): a new platform base image rebuilds the reference application from `main`, the rebuilt digest reaches dev, acc and every prd cell through the automated gates alone, and a rebuild whose tests fail stops in dev with an issue to the team ([section 1.5](#)).

Rebuild, backup and recovery

Test	Pass criterion
Rebuild the whole dev environment into an empty subscription from IaC + Git	Completes without manual steps; time measured and recorded as the environment RTO
Restore a deleted Key Vault secret and a deleted Key Vault (soft delete, purge protection)	Recovered; applications reconnect
PaaS point-in-time restore of the reference database	Within the agreed RPO / RTO
Stateful cluster (if built): restore of a zone's persistent volumes only (Azure Backup for AKS, vault tier) into the same and a rebuilt cluster, the pipeline deploying the StatefulSets afterwards	Data consistent; StatefulSets bind the restored claims; within the agreed RPO / RTO
Immutability of the backups: delete a recovery point, delete a blob version, shorten a retention and disable soft delete as <code>owner</code> of the backup subscription (section 6.3)	Every attempt denied or blocked by multi-user authorization
Region recovery drill: the primary region treated as lost; the environment rebuilt in the paired region from IaC, Git and the immutable copies only (section 6.3)	Reference application passes its tests; RPO per data store and RTO end to end recorded
Loss of the Flux source (Git unavailable)	Clusters keep running the last applied state; recovery when Git returns

Operations

- Dashboards and alerts for every cell, zone and dependency; every chaos experiment produced its alert, and the runbook linked from the alert worked.
- Telemetry: metrics, logs and traces of every cluster arrive in the environment's Grafana Cloud stack through Private Link, also from the stateful cluster; a trace of the reference application links the gateway, the

application and its logs; the absent-data alert fires when a cluster's Alloy is stopped ([section 6.2](#)).

- Logs: Kubernetes audit logs, Activity Log, Firewall and Traffic Manager logs arrive in the Log Analytics workspace with the agreed retention and can be queried from Grafana.
- No secrets in telemetry: a test application that logs a token, a connection string and an `Authorization` header; none of them reaches Grafana Cloud unredacted.
- Cost per environment measured at idle and at peak, and the prd estimate updated.
- Break-glass: the PIM break-glass path to a cluster and to Azure is used once end to end, raises its alert, and the operations agent opens the pull request that brings Git back in line. Repeated every six months in acc.
- Security detections ([section 6.2](#)): each one is triggered once on purpose and reaches its owner.