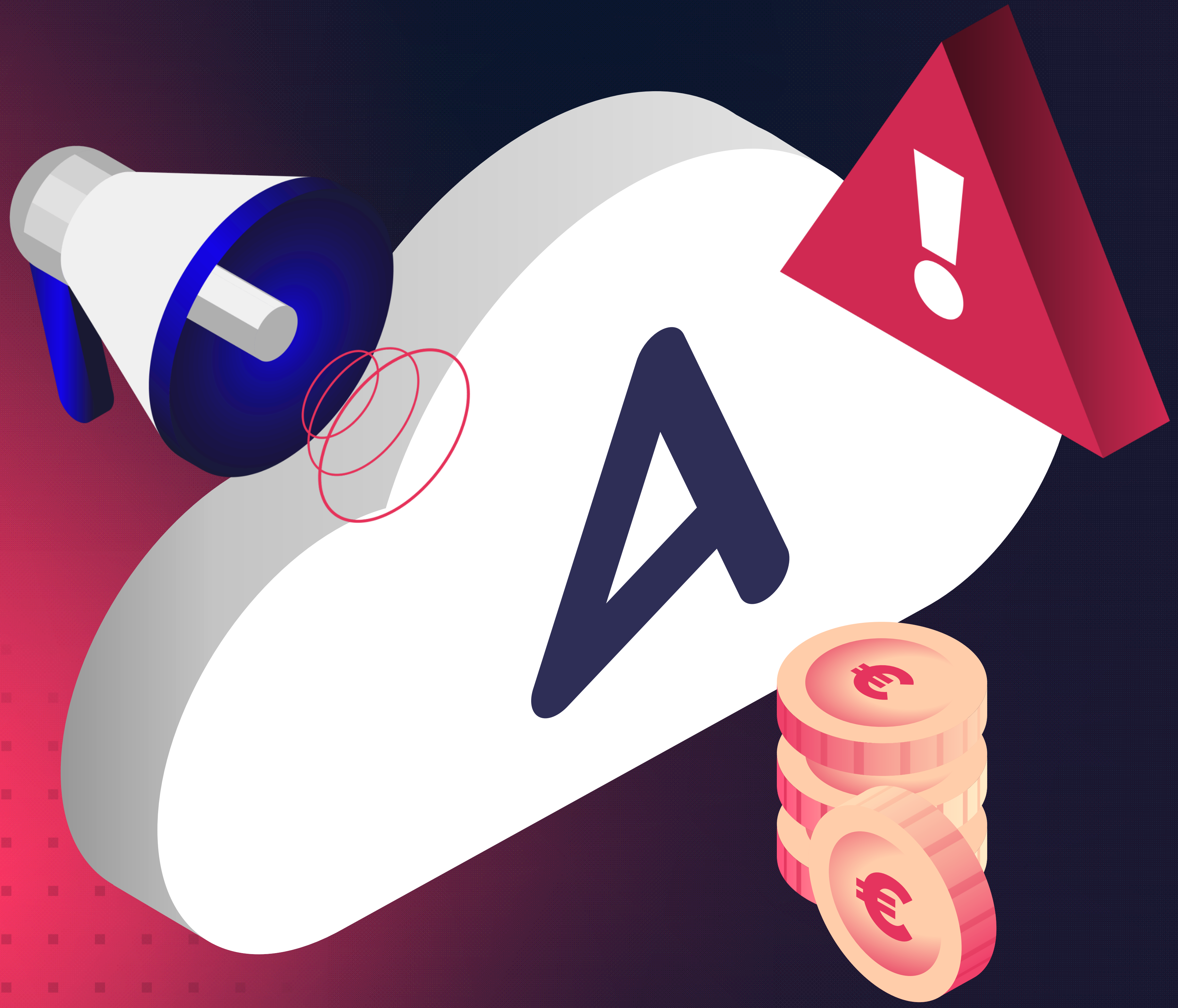
INTERCEPT

EBOOK



15 Biggest Mistakes in Azure

Avoid costly mistakes & learn what to do instead

Introduction

Welcome!

Microsoft Azure is a great cloud platform, but even small missteps can quickly turn into big problems.

At Intercept, we have been in the Azure game for years. Along the way, we've seen plenty go wrong and repeated poor decisions costing customers time and money.

That's why we compiled a list of the most common Azure mistakes so you don't ever make them (again).

Let's get started!

Mistake 1: Subscription

One subscription for everything

Using a single subscription for all your environments (production, staging, development, etc.) and applications causes more harm than good.

The solution

With just one subscription, it becomes hard to manage access, and it's easy to give developers too much control, which can lead to accidental changes or deletes in production. Azure offers tons of RBAC options, although they can't fully protect you if everything lives inside that one box: **the subscription**.

The solution? You probably guessed it; separate subscriptions per environment.

Organise resources

How you organise your cloud resources depends on your use case, though some steps apply across most scenarios. Setting it up correctly keeps your environment tidy, cost-efficient, and easier to manage :

1. Separate subscriptions per environment

Like Microsoft suggests, use separate subscriptions for development, testing, and production. This keeps environments isolated, reduces risk, and helps track costs per environment.

2. Resource Groups per app per environment

Create a Resource Group for each app in each environment. This keeps resources organised, simplifies access management, and makes monitoring and cost tracking easier.

3. *Service Groups to organise resources across subscriptions

Like fancy tags, they allow you to organise and group resources across subscriptions and resource groups (e.g., “Production apps”, “PCI Scope”, “MyApp”).

**Service Groups are a new feature as of May 2025. Learn more here.*

Be sure to follow these tips for your next app, and migrate existing resources along the way.

Mistake 2: ClickOps Nightmares

Clicking it all together

If you ever deployed resources in Azure, you know the Azure Portal is very intuitive and easy to start with. However, ClickOps doesn't scale and leads to more problems in the long run. Do you remember or document all the things you click?

As your systems grow, it quickly turns into a "mission impossible" to remember or document every click. The result? Mistakes and inconsistent environments.

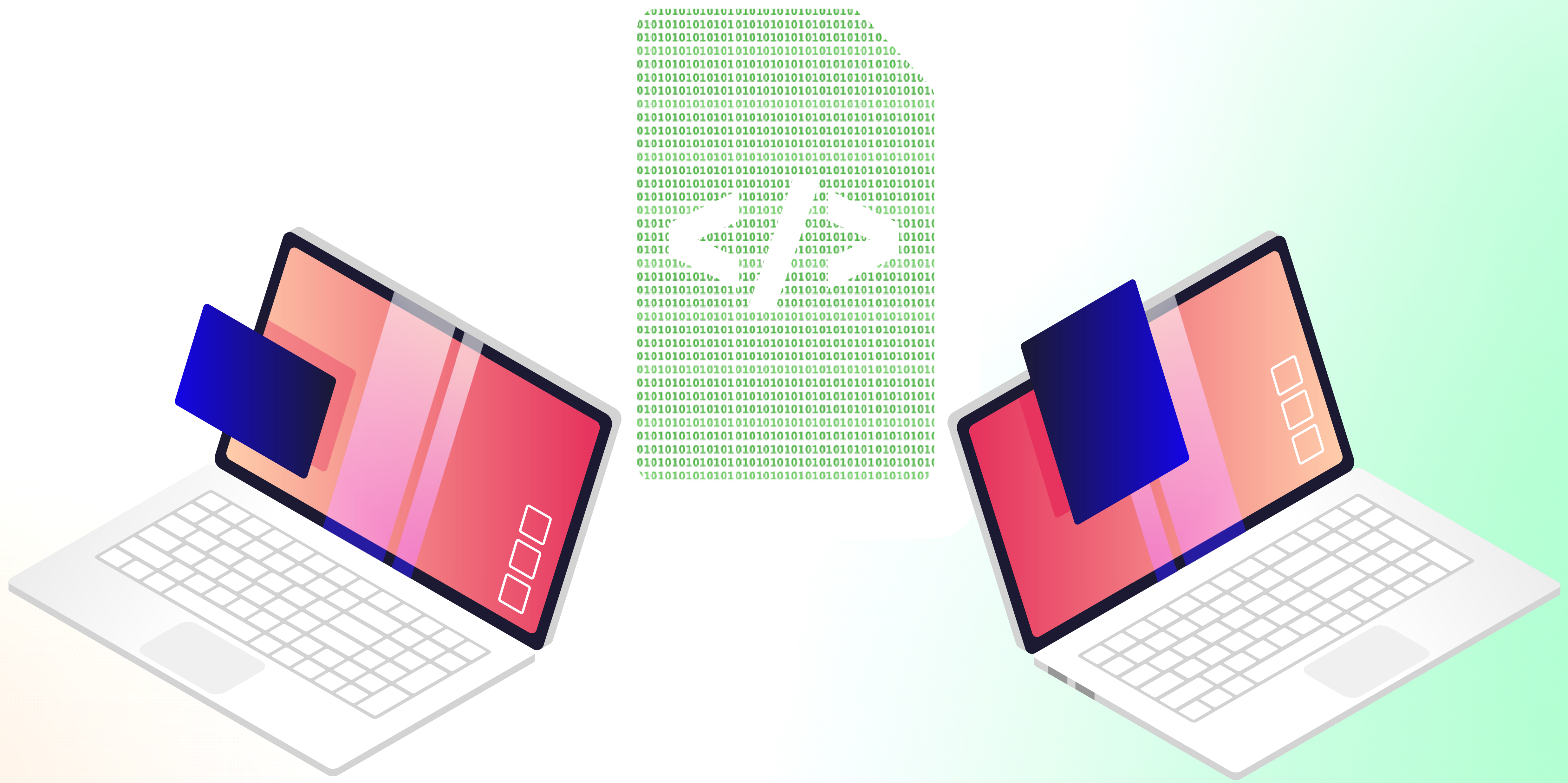
The solution

Infrastructure as Code (IaC)

Relying on the Azure Portal to deploy and manage infrastructure manually comes with risks and doesn't scale. Instead, move away from ClickOps to Infrastructure as Code; a more reliable and efficient way to manage your infrastructure.

There are many IaC tools widely available to achieve consistency, a single source of truth, and simplified documentation. All in all, IaC makes managing and scaling your infrastructure more efficient.

Relying on the Azure Portal to deploy and manage infrastructure manually comes with risks.



Do you want to know which IaC tool use for your workloads? Take a look at the differences between the available tools such as Azure Bicep, ARM, Terraform, and more here.

Mistake 3: Cost Management

Preventing expensive bills

Sudden, unexpected bills. You open your Azure invoice and blink. Did we really spend that much?

Let's see why and solve that!

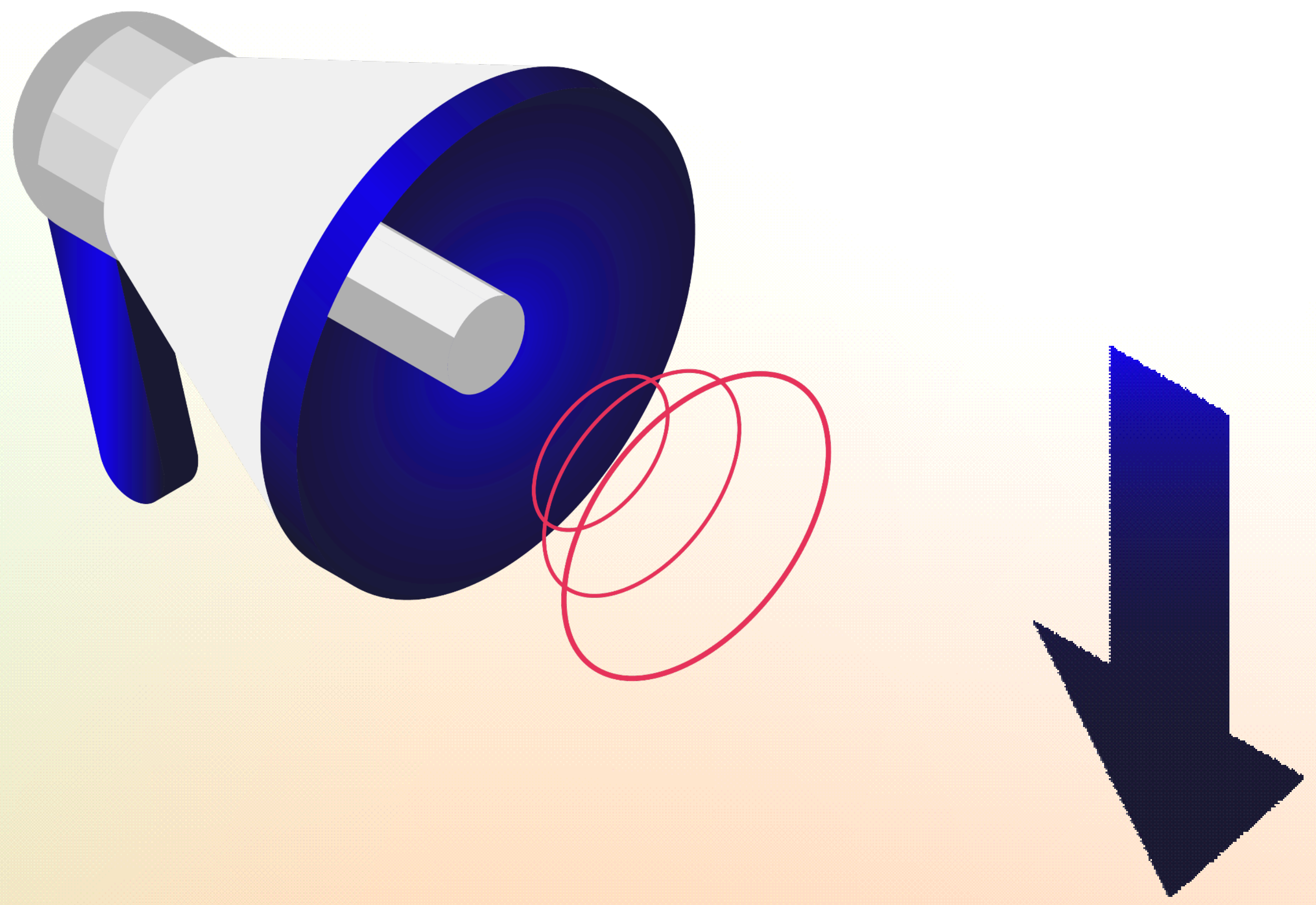
The solution

Your teams are deploying new resources, spinning up services; and when things are unmanaged, and there's:

- No early warning for forecast overrun.
- A lack of visibility across teams
- Missed anomalies and fraud detection

The result? Your Azure environment quickly spirals out of control, and costs skyrocket.

- Define a spending threshold for a subscription or resource groups.
- Set multiple thresholds (50%, 80%, 100%) to get early warnings.
- Track actual or forecasted costs over time; monthly, quarterly, or yearly.
- Trigger Action Groups (like email, Functions, or Webhooks) when thresholds are reached

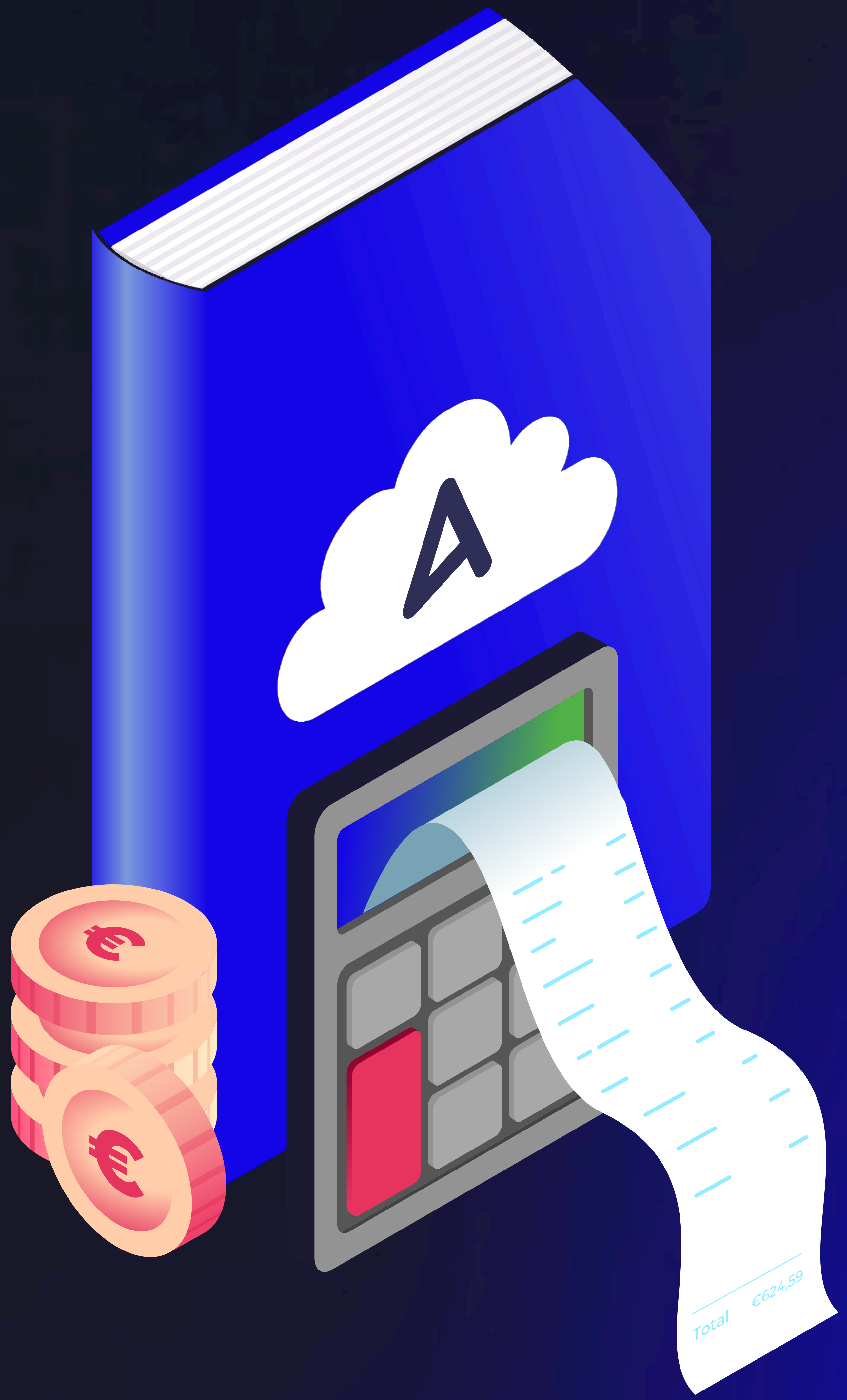


Out of all tips, this one is maybe the most important: Don't provision anything until you set up Azure budgets and alerts.

Learn how to manage and optimise costs in our **Azure Cost Management Whitepaper**, including Azure tools, discount methods and industry best practices.

Want a lower Azure bill? Check it out.

Download now



Mistake 4: Tags

Either forgotten or a big mess

As resources get added, managed by more people, it becomes tempting to forget “who made what”.

Which is why you need tags in Azure, which are metadata you apply to your resources. These key-value pairs let you label and identify resources based on attributes relevant to your organisation.

Inconsistent tags make it hard to answer questions like, for example:

→ How much does this app cost us per month?.

→ Who ownes this resource?

→ How much does the staging environment cost us?

The solution

Apply tags immediately from the start and implement them the right way.

Establish a tagging standard where you define a common set of tags for every resource like these examples:

→ **The environment (dev, staging, prod, etc.)**

→ **The owner (contact email)**

→ **The business unit (finance, sales, etc.)**

→ **Application (CRM)**

→ **Cost centre (global)**

→ **Service class (Tier 1, Tier 2, etc.)**

Mistake 5: Naming

Name it anything and regret later

When you first start your journey in Microsoft Azure, it's tempting to name resources however you like. But the problem is that soon our naming has no consistency across teams, apps and divisions. Consequently, it becomes hard to find stuff, slowing down troubleshooting. Not to forget about the added confusion, and the painful and challenging management of resources.

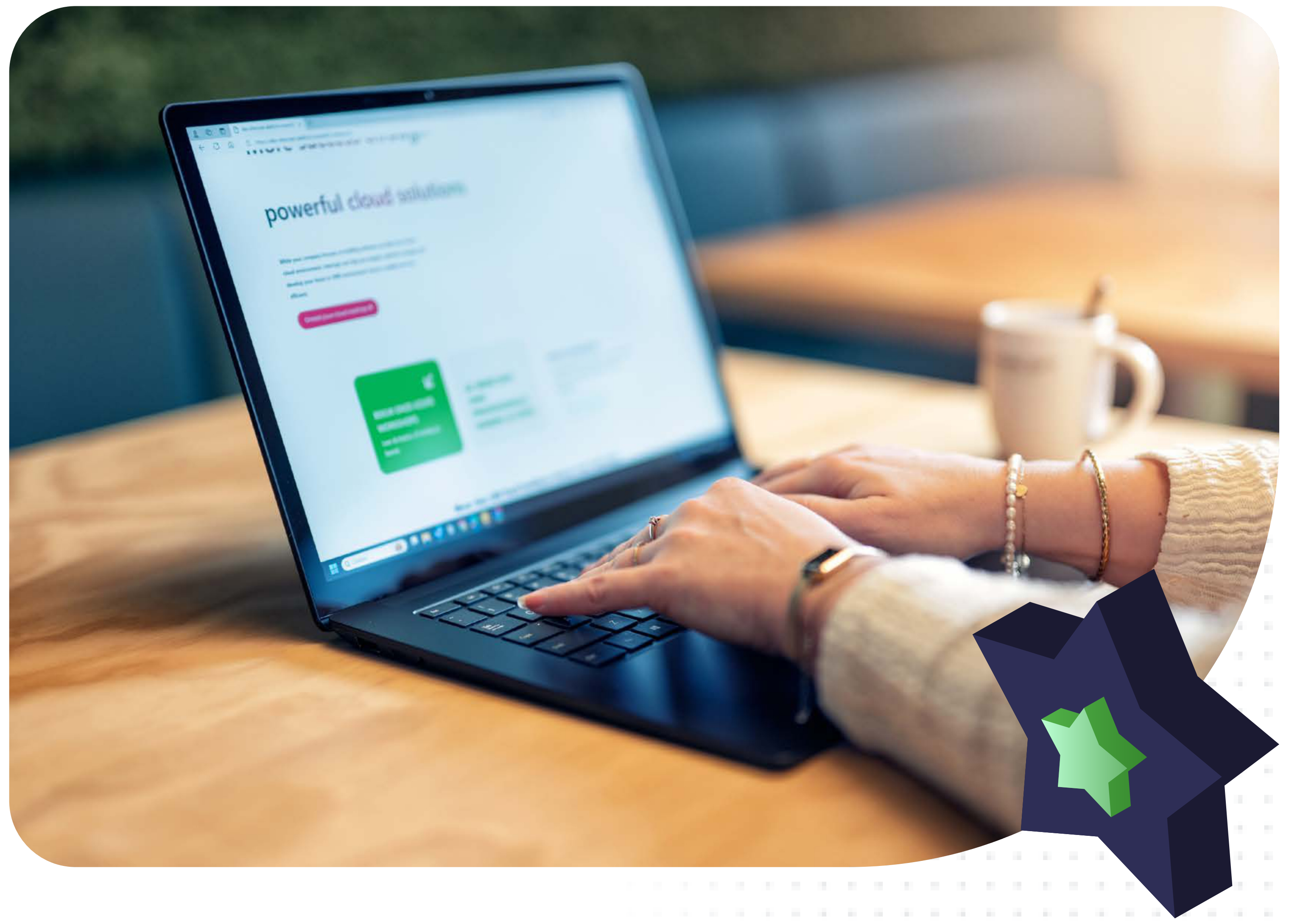
The solution

Implement a Naming Standard:

→ Follow Microsoft's recommended naming conventions and abbreviations. An example is rg-webapp-database-dev.

→ Use the Azure Naming Tool to generate consistent names.

The key here is to select a naming convection and stick with it along the road. And you can even enforce it using Azure Policy (which we explain next) for consistency across teams.



Note:
Not all resource names are allowed; like having hyphens in storage account names.

Mistake 6: Azure Policy

Standards gone rogue

Another common mistake is when organisations have all these documents defining their standards, yet no one follows it. Teams pick and choose what to apply, and before long, your environment becomes the “Wild West”. Without enforcement, naming conventions, tagging, etc. → configuration rules get ignored, resulting in a spreading disease of “cloud chaos”.

The solution

Use Azure Policies to enforce naming and tagging standards. Policies can have multiple effects:

Let's say you didn't restrict deployments by region or workload type. Now, someone deploys a resource in a high-cost region or spins up a VM much larger than needed. Before you know it, your bill has jumped, compliance requirements are at risk, and managing the environment becomes a nightmare.

→ **Multiple versions supported at once**

Prevent non-compliant resources from being created (preferred).

→ **Audit**

Flag resources as non-compliant.

→ **Modify**

Automatically change resources (not recommended).

However, enforcing policies aren't there only for stopping errors; it's also about setting up resources the way it should be like:

→ Enforcing HTTPS over HTTP

→ Turning on SQL Data Encryption at rest

Mistake 7: No monitoring or alerting

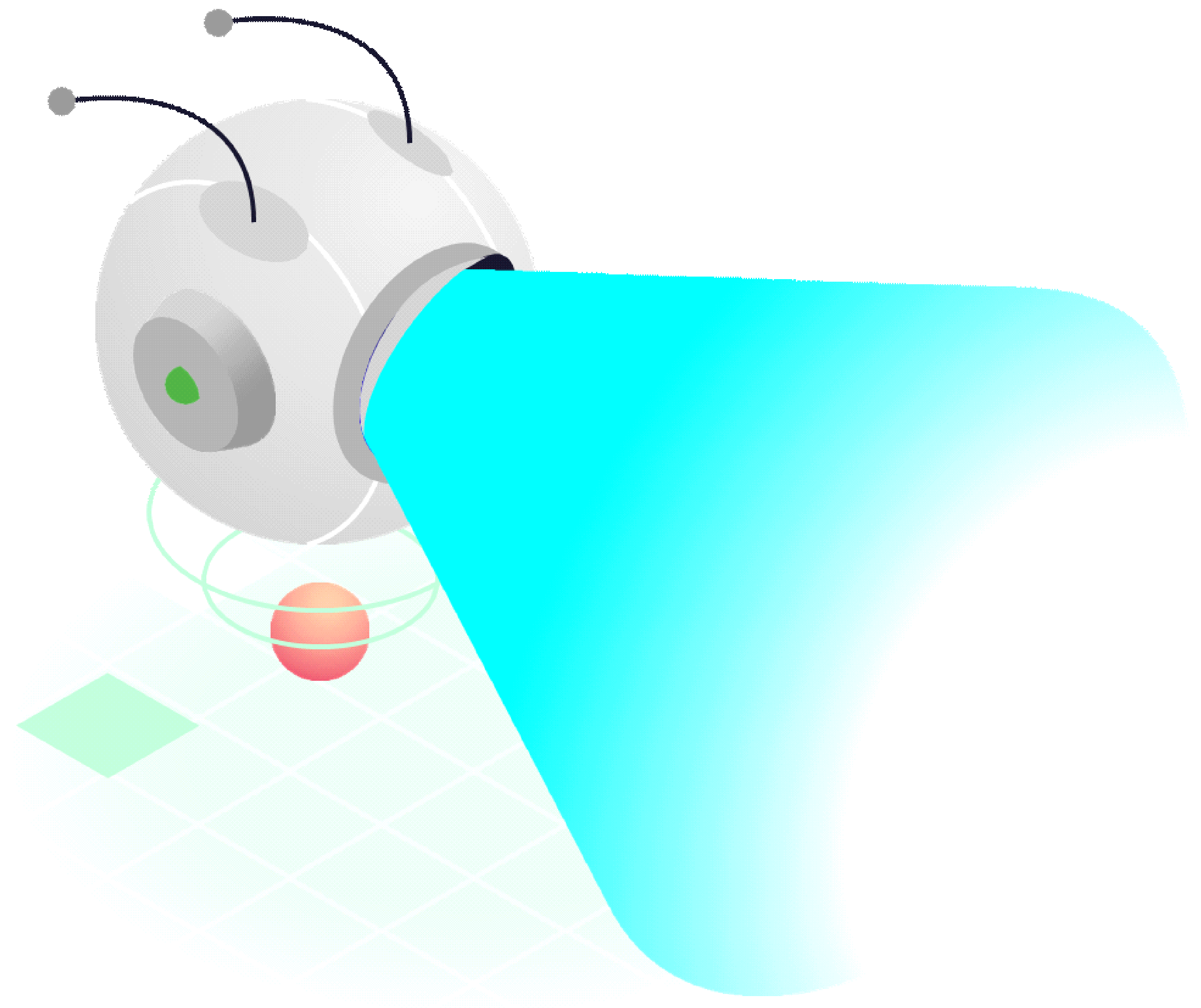
When everything looks “fine”

When you deploy your resources in Azure, you must not forget about monitoring and alerts. Because if something goes wrong, you might not notice until users do, and you would want that; it’s always better to catch issues before they affect your customers.

The solution

- **Monitor**
  Proactively monitor the health and performance of your applications and resources in Azure Use Azure native-monitoring tools such as Azure Monitor.
- **Alerts**
  Configure alerts for various metrics and logs that matter to your apps.
- **Notifications**
  Receive notifications via email, SMS, or other channels.

This lets you act quickly and proactively, keeping your applications healthy and users happy.



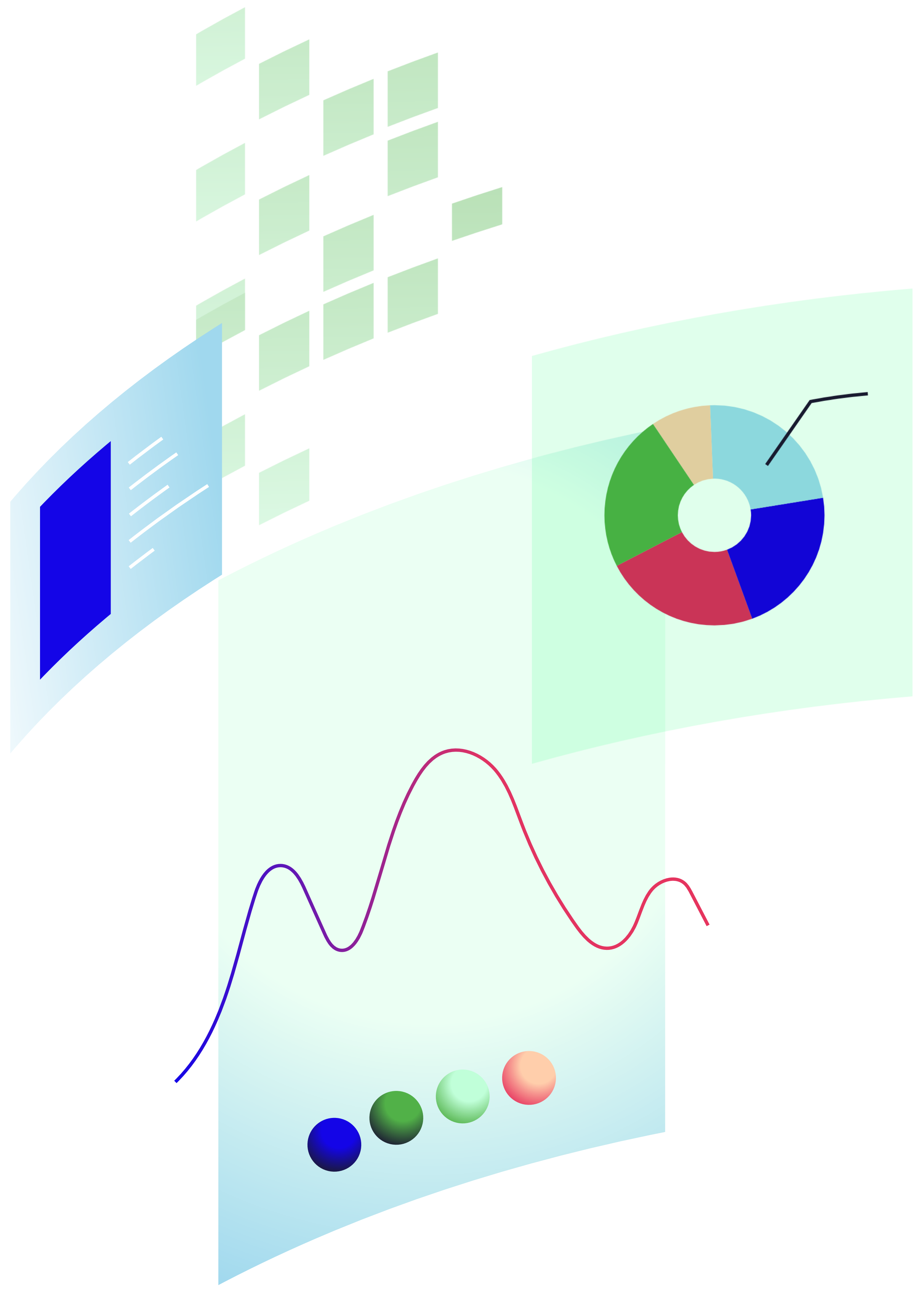
Mistake 8: Unfiltered log ingestion

Ingest first, regret later

We see this mistake occurring often with developers: sending more than needed to Log Analytics. If all logs are sent straight to Log Analytics, sensitive information can slip through and the volume of data can explode quickly. By default, everything gets ingested, which can lead to costs you would never think of.

The solution

- Sample log data to reduce volume; either before sending to Application Insights (recommended) or during ingestion.
- Set a daily workspace cap in Log Analytics to prevent unexpected spikes in costs (great during high app activities or many errors). Note that once the gap you set is reached, additional logs aren't captured, leading to data loss.
- Also follow Microsoft's recommendations for logging.



Mistake 9: Idle resources

Zombie resources

It’s easy to spin up resources in Azure, overprovision them or even forget that they exist...Orphan resources are those created by someone “once upon a time”, yet nobody knows who.

The issue

Sometimes resources get left behind when part of a larger system is removed. The usual culprits are:

- IP addresses no longer attached to machines
- Virtual Machines (VMs) that aren't deallocated
- Unattached disks that continue to incur costs
- Deleted VMs with backups that haven't expired; which you keep paying for
- Empty subscriptions and resource groups.

Orphaned resources don't always cost you money, but they can. At best, they create confusion; at worst, they become a security risk.

The solution

- ✓ Use Azure Cost Advisor to find orphaned resources underutilised or any other idle resources.
- ✓ Use Azure Resource Graph to query for resources that aren't attached or are in an unexpected state.
- ✓ Regularly clean up orphaned resources to reduce cost, improve security and keep your environment clean.

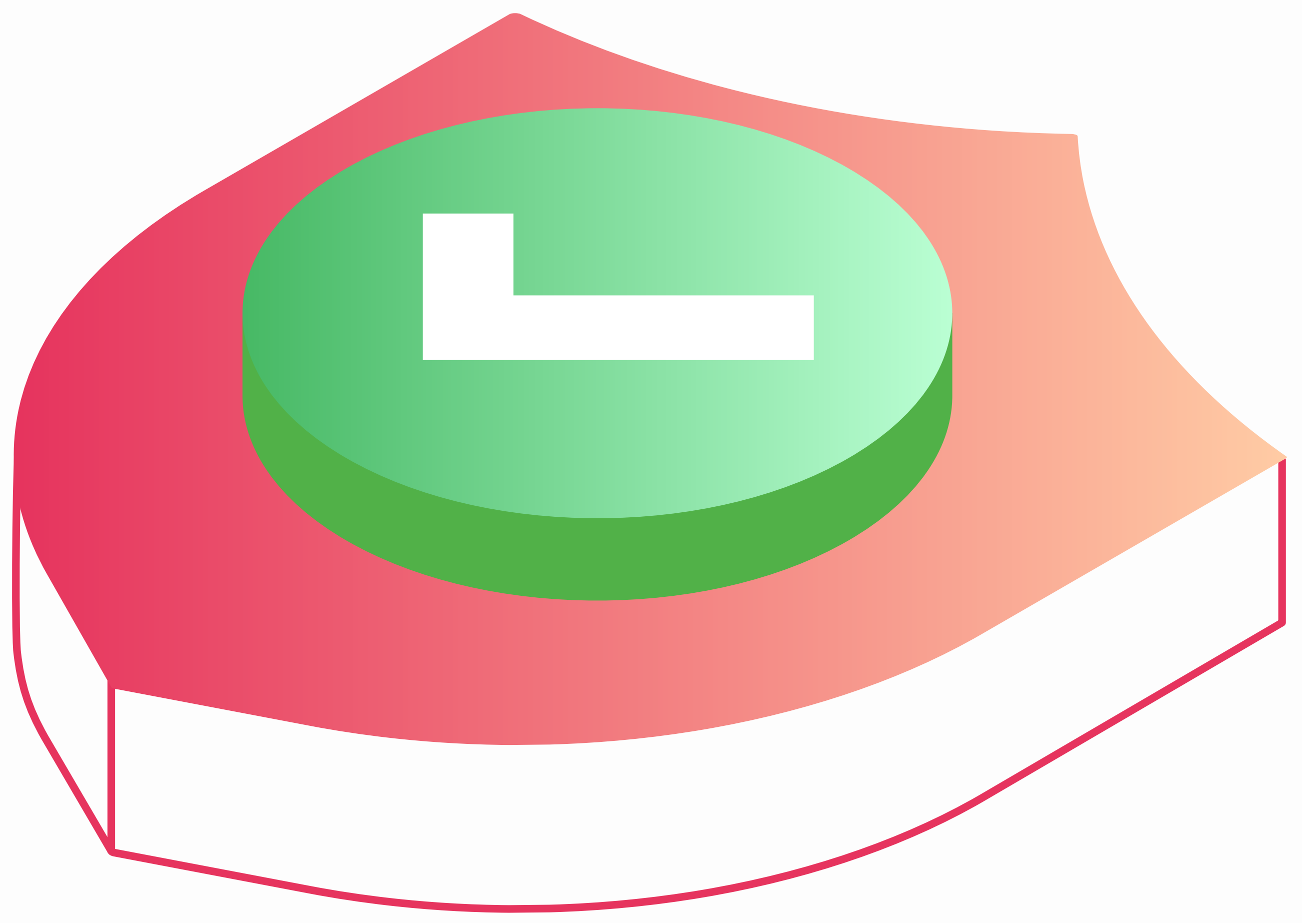
Mistake 10: Not having RBAC Strategies

Too many hands, too much power

Another mistake we often see is giving everyone high privileges in Azure, ending up with owners all over the place. What started as convenience now turned into risk. And instead of a least-privilege model, you've got an environment where a single wrong click can take down production.

The solution

Design your solution with the least privilege in mind, following a Zero Trust approach where no one is presumably trusted. Start with built-in roles in Azure and only grant permission that's needed for the job.



→ Use role assignments at the right scope (management group, subscription, resource group).

→ Limit Owner and Contributor roles to a few trusted admins.

→ Create custom roles if the built-in ones are too broad

→ Review role assignments regularly and remove what's no longer needed.

→ Use Privileged Identity Management (PIM) for time-bound access instead of permanent rights.

Mistake 11: Managed Identity

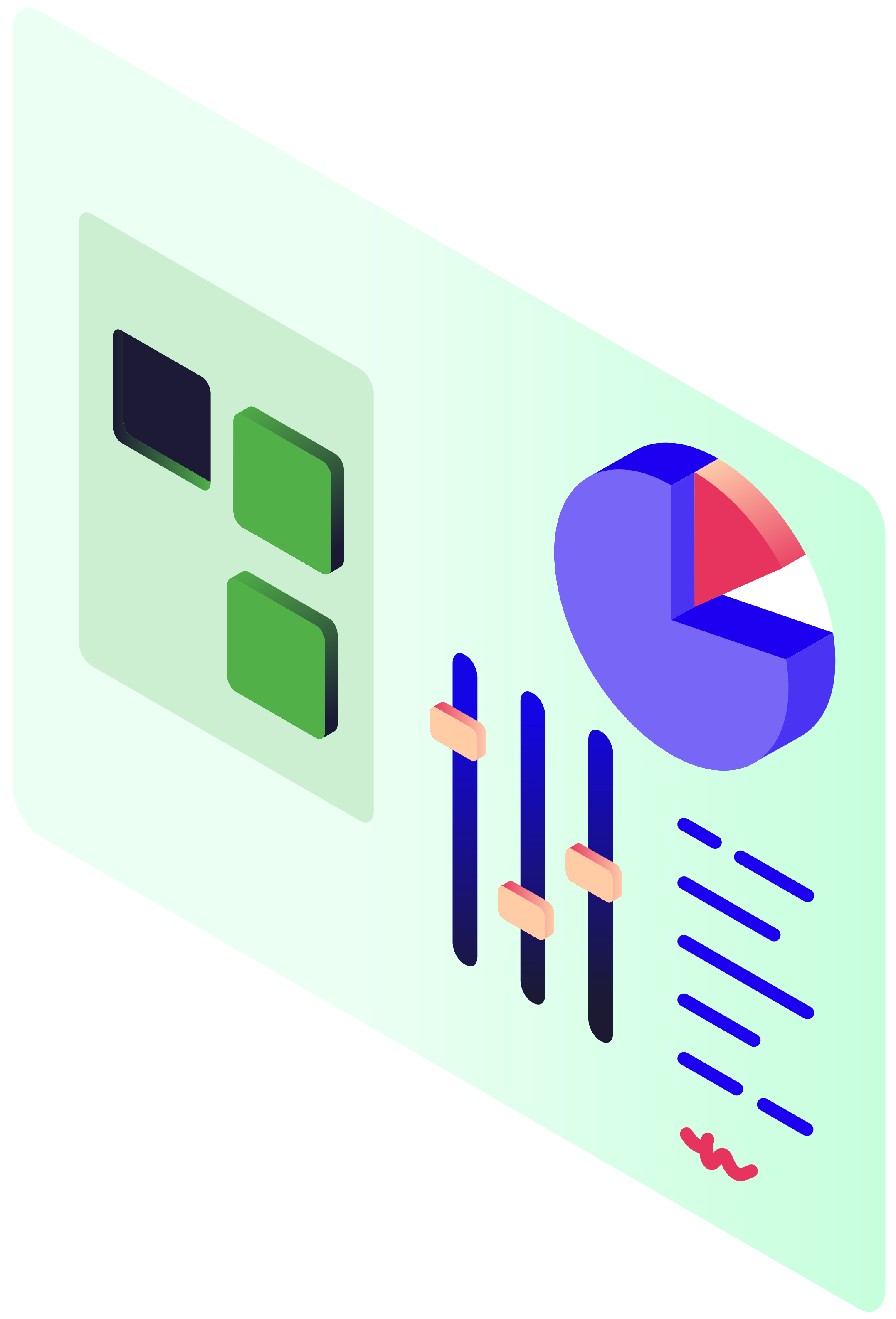
Anything but the safe way

Too often, we still see teams juggling credentials. You create, store, rotate and revoke them, until someone forgets. Azure SQL is still accessed with plain SQL Authentication (username + password), and Key Vault with a client ID and secret. Sooner or later, someone commits a secret by accident.

The solution

Use Managed Identities in Azure. These allow you to let a resource run with its “own” identity, managed entirely by Azure. No usernames, passwords, client IDs, or secrets to worry about.

- Assign a Managed Identity to an App Service, Function, or other resource.
- Grant that identity the right access to read secrets, connect to databases, and more.
- Completely passwordless.
- In your code, use **DefaultAzureCredential** to handle authentication automatically.



Mistake 12: Not using KeyVault references

Copy-paste credentials

Too often, we still see teams treating secrets like configuration. Key Vault exists for a reason, yet credentials end up in env vars and config files, until someone pushes them to source control or shares them with anyone who can read the app settings. Over time, that “temporary” shortcut becomes a serious security risk.

What is Azure Key Vault?

Azure Key Vault is a SaaS service for securely storing and accessing certificates, keys, and secrets.

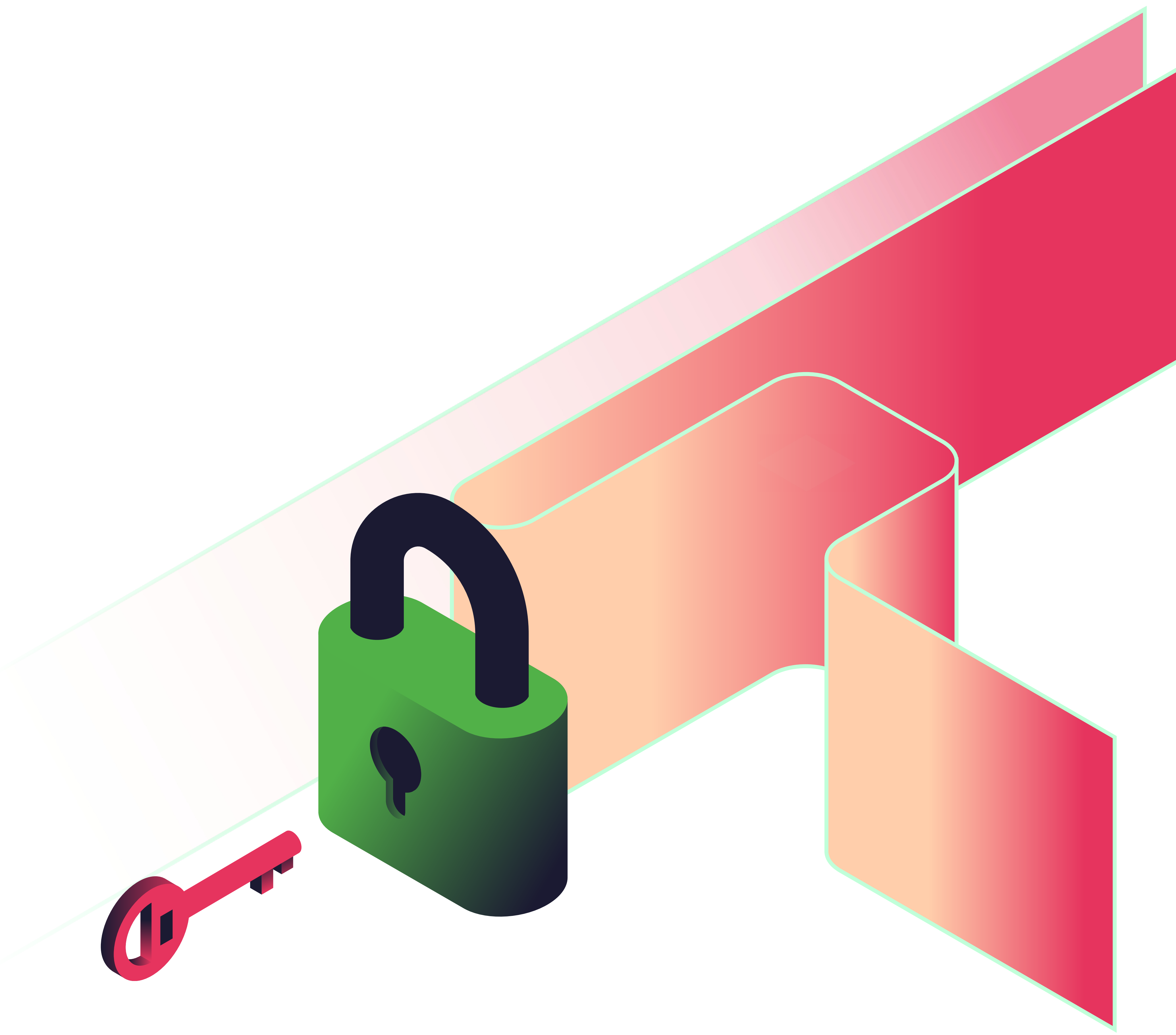
We can safely store confidential data such as passwords (called secrets in Azure), cryptographic keys, certificates, and encryption keys in a centralised place.

Nonetheless, we still see teams leaving secrets in environment variables or configuration files. This can lead to accidental exposure, secrets being committed to source control, or credentials being accessible to anyone with access to app configuration. Over time, it becomes a serious security risk.

The solution

Use Key Vault References to securely point to secrets from locations without storing them directly. This keeps sensitive data out of code and app settings while allowing applications to retrieve secrets at runtime.

- **Can be used with App Services, Functions, and other Azure Services.**
- **Simplifies secret rotation; update the secret in key Vault, and apps automatically get the new value.**
- **Significantly reduces the risk of accidental exposure or commits.**



Mistake 13: Poorly designed Networks

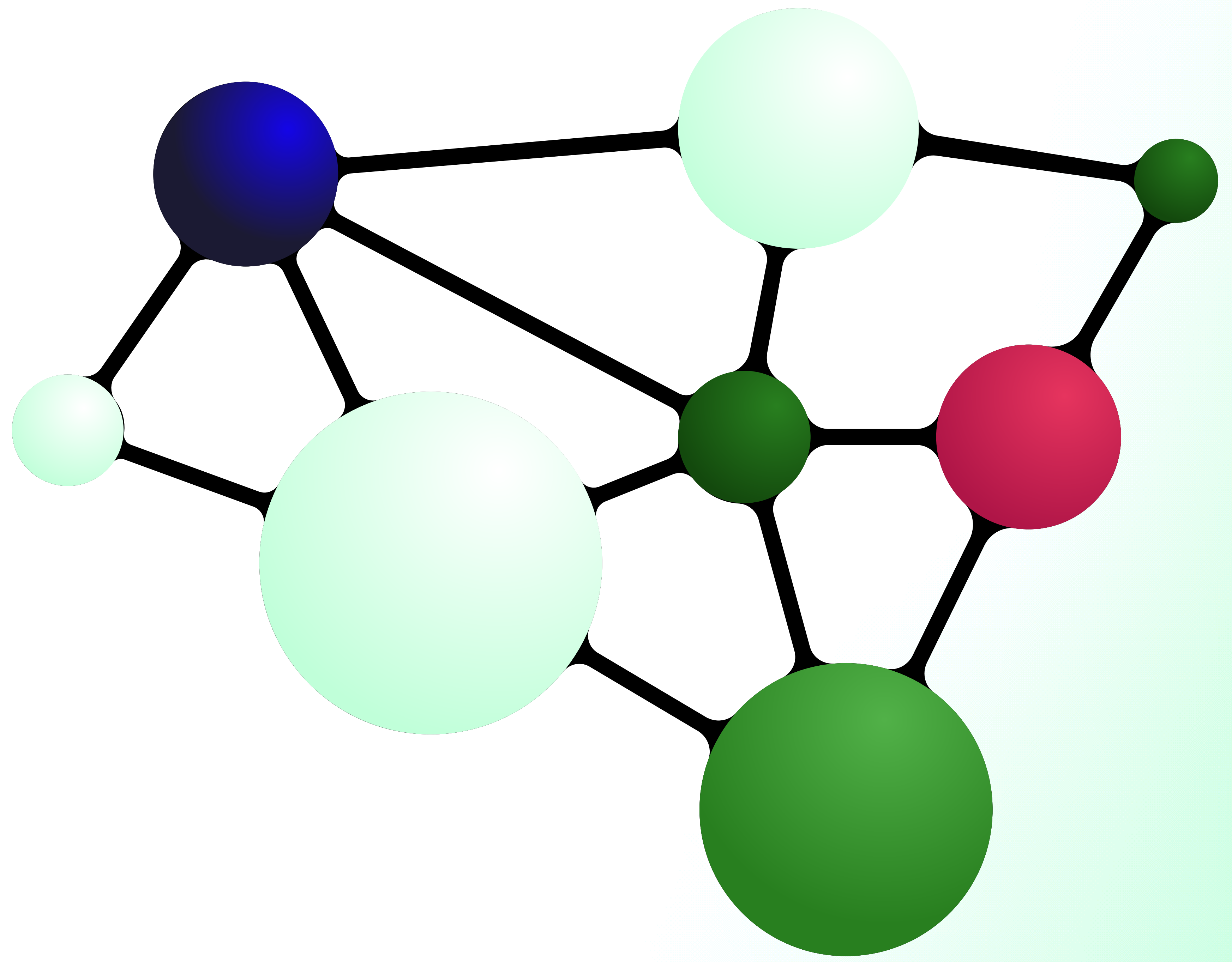
Doors left wide open

This one's more of an architectural mistake, but we've seen one too many flat network topologies in Azure. Trying to manage it almost feels like you're stumbling through a maze in the dark.

The solution

A better way is to design a hub-spoke model with a firewall: a central point of control and the ability to filter or block unwanted traffic.

- ✓ **Central control**: One place for ingress/egress and shared services.
- ✓ **Consistent filtering**: Inspect, allow/deny, and log traffic centrally.
- ✓ **Strong segmentation:** Isolate apps/ environments and reduce blast radius.
- ✓ **Predictable routing:** UDRs force traffic through the firewall/NVA.
- ✓ **Standard patterns:** Repeatable spokes per workload/subscription.
- ✓ **Better visibility**: Clearer flow logs and faster troubleshooting.



Net result: your network becomes easier to manage, more secure, and far less chaotic, because it's designed like a system, not grown like a patchwork.

Mistake 14: Overprovisioning resources

Scale by guessing

Allocating more resources than you actually need is an easy way to waste money. We've seen plenty of customers who aren't sure which SKU they really need, so they give VMs extra CPUs and memory "just in case" and end up paying for unused capacity. Add manual scaling on top of that, and the bill grows fast.

The solution

Follow these best practices:

Best practice 1: Authentication best practices

- **Right-size first:** Begin with the smallest SKU that meets baseline needs.
- **Enable autoscaling:** Let demand drive capacity, not guesses.

Best practice 2: Choose the right VM

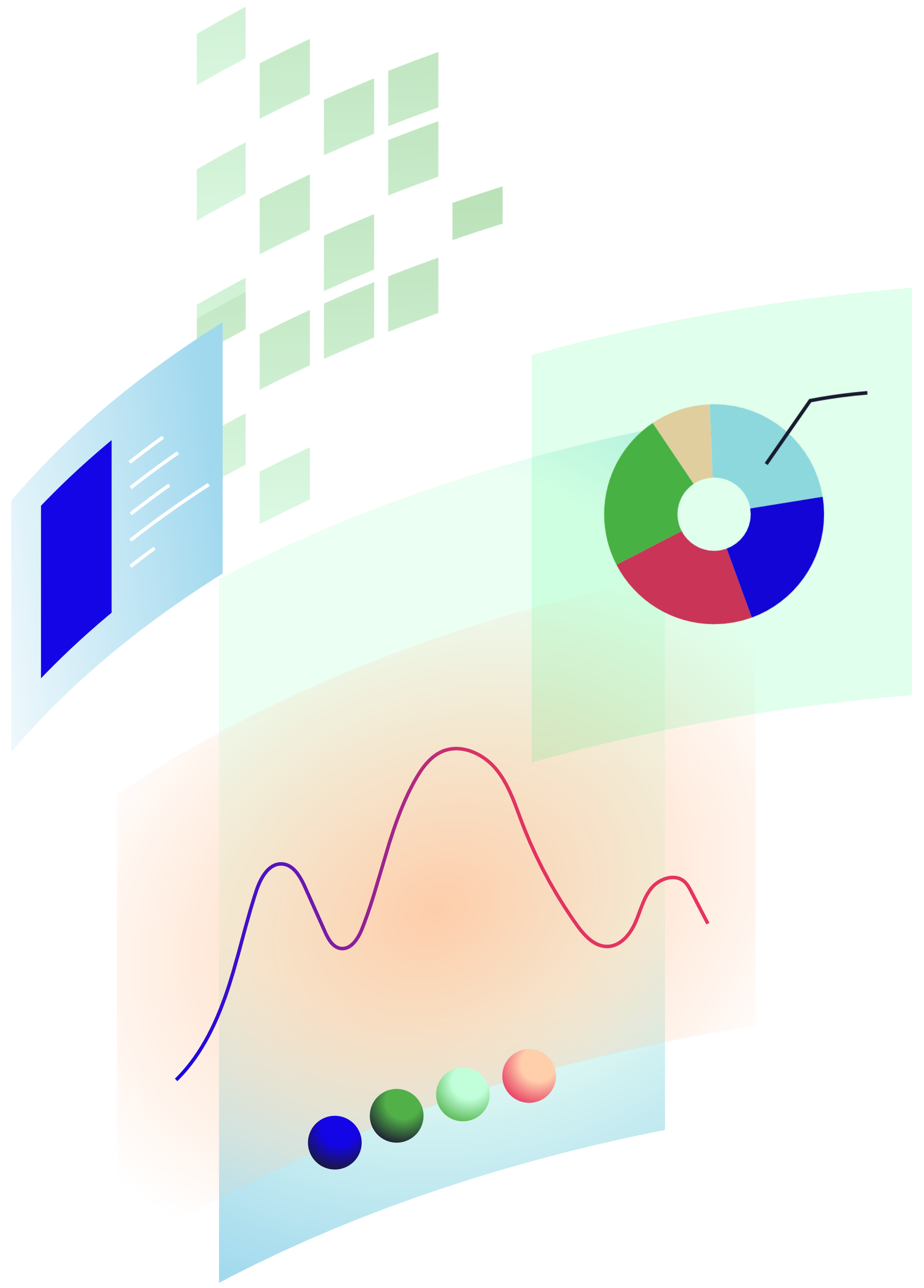
- **Know your workloads:** Analyse what each application needs in terms of CPU, memory, throughput, and network.
- **Use heatmaps:** Visualise usage over time to spot spikes, trends, and persistent waste.
- **Test different configurations:** Compare smaller and larger sizes to find a cost-effective sweet spot.

Best practice 3: Use VM Scale Sets

- Define clear scale rules (metrics + thresholds).
- Combine with load balancing so traffic is spread evenly.

Best practice 4: Use Azure Advisor

- **Review recommendations regularly:** Right-sizing and cost suggestions change as workloads change.
- **Track savings:** Turn quick wins into ongoing hygiene.



Note:
Right-sizing means matching resources to your needs, so you only pay for what you use. A practical place to start is identifying underused instances (for example ~20–30% CPU utilization) and resizing them down — this can often save 10–15% on the monthly bill.

Mistake 15: Backup and disaster recovery

Hope is not a plan

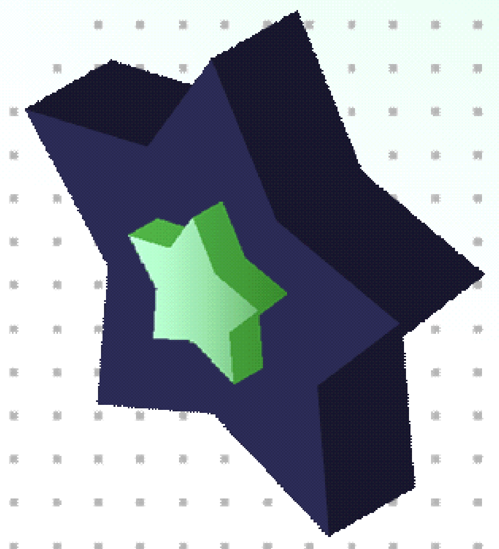
Ask yourself:

- What happens if a natural disaster hits your Azure region?
- What are the consequences if your environment goes down, or customer data is lost?
- How long can your business operate without access to critical systems or data?

If you're unprepared, any disaster can directly affect business revenue or even take you out of business.

The solution

- Use Azure Backup to protect your critical data. It allows you to create and restore backups of your data and resources, to protect your environment against accidental deletion, corruption, or malicious attacks.
- Use Azure Site Recovery to replicate workloads and ensure business continuity in case of an outage.
- Consider geo- and es zone-redundancy: replicate data across regions (GRS, GTM, ASR) and distribute workloads across availability zones (ZRS, zone-redundant services) to protect against regional or zone-level failures.
- Treat backup and disaster recovery in Azure as a routine part of your environment; by testing regularly.



Final thoughts

Azure offers endless and outstanding possibilities, but managing Azure comes with plenty of pitfalls. Some of the outlined common mistakes are risky and costly, but you now know how to avoid them and what to do instead by using the best practices, which in turn can lead to:

- ✓ **Lower overall Azure costs**
- ✓ **Less manual effort for security and account management**
- ✓ **Stronger security posture in the cloud**
- ✓ **Improved application performance**

And the list goes on...

INTERCEPT

Which mistakes exist in your environment?

Request a Well-Architected Framework Scan and receive clear, prioritised recommendations across the 5 pillars, aligned with Microsoft's best practices. Apply them to reduce risk and errors, improve reliability, and control cloud costs, so you get more value from the cloud as you scale.

Request your free scan now!

Azure Workshops

Interested to learn more about Azure? We have multiple free online Azure workshops to help you understand the ins and outs of Azure.

→ View all workshops