Engineer

Azure

Microsoft's cloud platform

What it is

Azure, or Microsoft Azure in full, is a catalogue of computing rented out from Microsoft's own data centres: virtual machines, storage accounts, managed databases, container hosting, private networks and the tools to run them, paid for by use. Microsoft sold it as Windows Azure until April 2014, when it took its present name.

Every request to manage Azure resources, whether a click in the portal, an Azure CLI or PowerShell command, an SDK call or a template, arrives at the same front door: Azure Resource Manager, which authenticates and authorises it and hands it to the service that does the work. A virtual machine, a database server and a DNS zone are all resources in its sense, each with an ID, a type and a set of properties.

Resources are filed in a hierarchy four levels deep. Each resource belongs to exactly one resource group, a folder meant for things that are deployed, updated and deleted together; deleting the group deletes everything in it. Resource groups live inside a subscription, which is at once a billing unit and a boundary for access and limits. Subscriptions are gathered into management groups, which can nest up to six levels and all fold up to a single root management group for the directory. Role assignments, policies and budgets set at one level reach everything beneath it.

Over the whole hierarchy sits identity. An organisation's users, groups and applications live in a Microsoft Entra ID tenant, the directory Microsoft called Azure Active Directory until 2023, and every Azure subscription trusts exactly one such tenant to sign people in and decide what they may do.

Use the concept in a decision

Put “What is Azure?” to work in a real decision. We can help relate the concept to the system you are building or buying. Servers sized for what you actually use, moved into the cloud or back out of it, with the monthly bill worked out first.

Explore our services Get in touch →

Test the assumptions behind the term

Use the explanation and linked work to frame questions for your project. We can examine the assumptions with you before a tool is chosen. Ran 20 Docker apps on Google Cloud and AWS — 20 containerised applications deployed and watched

Ran 20 Docker apps on Google Cloud and AWS Get in touch →

Why it matters

Most teams that run on Azure got there through something they already had. A company whose staff sign in to Microsoft 365 already keeps its people in an Entra tenant, and a subscription that trusts that tenant can give the same accounts access to servers and databases without a second user directory to maintain. For a supplier, the choice of Azure therefore turns less on feature tables than on where the client’s identities and data already sit.

The hierarchy is the second reason, because governance lives in it. Microsoft’s guidance on organising resources recommends separate subscriptions for development, testing and production, so that environments cannot grow dependencies on each other and each can carry its own policies and its own line on the bill. Azure Policy then enforces house rules at whatever level it is assigned. It can deny a change, such as a resource created in a region the organisation does not use, or log it for someone to review, and Microsoft advises starting a new policy in audit mode before letting it block anything.

The third reason is that Azure can be described rather than clicked. Resource Manager accepts declarative templates, written in JSON or in Bicep, and works out for itself what to create or change. Running the same template twice leaves the same result, and Azure stores the state of what was deployed, so there is no separate state file to look after. An estate written this way can be reviewed as a pull request, rebuilt after a bad day and copied to a new environment by changing its parameters, which is the main defence against an Azure setup that only one person understands.

What it looks like in practice

The tech stack page names Google Cloud and AWS as the company's usual clouds, with Azure where a client already runs there. Two portfolio entries show that Azure work from both ends.

On the infrastructure side, three Azure environments were written down as code, all from the same Bicep templates: the api and www as Azure Container Apps on a managed environment, a PostgreSQL Flexible Server, Redis for caching, Front Door with a WAF policy in front, and a VNet, NSG and private DNS underneath, with Log Analytics wired in for diagnostics and images pulled from the project's Azure Container Registry. Because the templates are parameterised, a new environment or a change to an existing one is a pull request, and drift stopped being a mystery, since the code is the source of truth.

On the release side, deployment to Azure was owned end to end. Every release travels development, staging and production in that order, and nobody pushes straight to a live environment. CI/CD builds and ships the images, while the templates hold each Azure environment to the same shape, so a promoted build never lands somewhere subtly different. Settings that vary between environments are stored separately from secrets, which is what lets one built artefact pass along the whole route and pick up its own settings on arrival, with no ad-hoc manual deploys along the way.

Where it goes wrong

The costly Azure mistakes tend to share one shape: a gap between what someone believes Azure did, or allowed, and what Resource Manager actually did.

A template deploy leaves things behind and puts things back. Resource Manager deploys in incremental mode by default. In that mode, a resource that exists in the resource group but is missing from the template is left alone, so deleting a database from the template file leaves the server running and billing. The same mode reapplies every property of each resource it redeploys, and a property the template leaves out is reset to its default. A setting someone changed in the portal is therefore overwritten on the next routine release. The what-if preview can warn of both, though Microsoft notes that its output may contain noise, false positives among which a real change is easy to miss. The

template should state every value that differs from the default, and a deployment stack, which detaches or deletes the resources that drop out of a template, handles the leftovers.

Every environment shares one subscription. It is the quickest way to start: one subscription, a resource group per environment, and broad roles granted at the top so nobody is blocked. Production and test then share role assignments, policies, limits and a single bill, and a script aimed at the test group runs with rights that also reach production. Separate subscriptions under a management group give each environment its own access, rules and costs.

A directory admin is taken for a resource admin. Microsoft Entra roles, such as Global Administrator, govern the directory: users, groups, passwords and domains. Azure roles, such as Owner and Contributor, govern resources. By default neither set reaches into the other. A Global Administrator can flip a switch that grants them User Access Administrator at the root scope, which lets them view every resource and assign access in any subscription in the tenant. The switch exists to regain access to a subscription, and Microsoft's guidance is to remove the elevated access once the work is done. Left in place, it is a directory account that can hand itself any resource.

A container app that never sleeps. Azure Container Apps can scale most applications to zero replicas when nothing is calling them, which is part of its appeal for small services. Applications that scale on CPU or memory load are the exception and cannot scale to zero. An app given a CPU rule keeps at least one replica running through every quiet night, so a service chosen for idling cheaply should scale on HTTP traffic or another event source.

Further reading

- Azure documentation — Microsoft's index of guides, quickstarts and references for every Azure service.
- What is Azure Resource Manager? — the layer management requests pass through, and how to draw resource groups.
- Organize your Azure resources — the Cloud Adoption Framework on management groups, subscriptions and resource groups.
- Deployment modes — what incremental mode leaves running and which properties a redeploy resets.
- Deployment stacks — managing a template's resources as one unit, including those removed from it.

- Azure roles and Microsoft Entra roles — where directory roles end, where resource roles begin, and how elevation joins them.

Glossary

Choose for your actual constraints

Bring the deployment path, the operational risks and the people on call. We can scope a change with checks, a recovery route and clear ownership.

See what it costs Get in touch →

Make the terminology practical

Ask how “What is Azure?” applies to your organisation. A concrete example from your current system is enough to start the discussion.

Explore our services

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://engineer.company/glossary/azure/