Microsoft Azure

Migrate and Modernize with Linux on Azure

Building a unified, secure and AI-ready environment for Linux apps and open-source databases

Contents

Get started with Linux on Azure

Across industries, enterprises are running business-critical workloads using Linux on Azure and open-source software (OSS) to do more with less – keep critical services reliable, meet rising security and compliance expectations, control costs, meet demand spikes and ship features faster across hybrid or multicloud environments.

As data and customer expectations outpace on-premises capabilities, legacy footprints and fragmented tool chains make even routine changes risky and costly. Managing the operational drag of multiple Linux distributions, each with its unique quirks, slows business responsiveness at a time when regulation, cost pressures and competitive advantage demand more agility and security than before.

The answer is to bring your Linux apps and OSS databases together on a unified platform that provides consistent security, governance and operations – so you can move faster with less risk.

A strategic approach to migration and modernisation spans infrastructure, platforms, applications, data and teams. You address immediate needs while paving the way for future-looking initiatives. The business impact of Linux on Azure comes together with faster release cycles from standardised delivery, lower total cost of ownership (TCO) from managed data and platform services and improved resilience and security through consistent baselines.

About this eBook

This eBook is your practical on-ramp to cloud-native innovation and AI readiness. It begins by showing how to choose a migration and modernisation strategy for your Linux applications and OSS databases and bring them together onto a unified, governed Azure platform. By following this guidance, you can make informed migration decisions with governance and optimisation in mind – reducing sprawl, lowering total cost of ownership and gaining the agility to prepare for what's next.

Reduce complexity with a cloud-ready strategy

Managing a diverse Linux estate across multiple distributions and database platforms is a recipe for complexity.

Inconsistent environments are harder to patch, secure and automate – especially at scale. Each distribution brings its own package managers, kernel versions, security baselines and update cycles, making standardisation a challenge.

By defining a cloud-ready strategy upfront, you can unify operations, enforce consistent security and data practices and simplify governance. Whether you migrate via VMs, containers or cloud-native services, this approach reduces sprawl and sets the foundation for continuous modernisation.

Assess, plan and get ready

Successful Linux modernisation is less about a single migration event and more about a disciplined sequence of moves that reduce risk and accelerate delivery:

- **Assess your IT estate, identify critical Linux apps and OSS databases for modernisation and prioritise.**
- **Make a plan that aligns cloud migration options with business goals, compliance and risk tolerance.**
- **Launch your strategy.**

Your business priorities shape the right technical strategy. For example, to lower TCO, you can shift capital expenditures (CapEx) to operational expenditures (OpEx) by adopting pay-as-you-go cloud Infrastructure-as-a-Service (IaaS). To add room to tight IT budgets, you can move to Platform-as-a-Service (PaaS) and reduce licensing and operational overhead.

Working in phases, you can effectively navigate complex moves while delivering business value at each stage. Strategic platform choices help you manage risk and governance and future-proof your infrastructure while providing familiar open-source operating systems (OSs) and tools.

Accelerate migration and define where you want to land

To reduce risk and accelerate migration and modernisation, start by deciding where your workloads will land. A landing zone provides the cloud foundation and governance framework you need to migrate with confidence. Landing zones are enterprise-wide guardrails. They establish a consistent, policy-driven environment for your Linux and OSS workloads, so you don't have to start from scratch or reinvent compliance controls with every workload you migrate.

Implemented as Infrastructure-as-Code (IaC), landing zones clarify critical design decisions and simplify networking, identity and security. They provide ready environments for any modernisation path you choose for your apps and data.

Considerations

- **Establish a strong baseline.** Define which Linux distributions and OSS components are approved for use in your environment, set security and patching standards and align your landing zone with enterprise policies.
- **Prepare your data for modernisation.** Validate data quality and governance and plan for real-time analytics and AI. For sensitive workloads, keep data protected even while in use with confidential computing options such as Azure Confidential VMs and Azure Kubernetes Service (AKS) confidential containers.
- **Build security and compliance in from the start.** Enforce encryption (at rest and in transit), centralised key management (avoid hard-coded secrets via managed identities) and policy-as-code so every migrated or modernised workload is compliant by default.

Rationalise before you migrate and modernise

Moving to the cloud is an opportunity to rationalise your Linux estate and assess the unique needs of each workload. With an inventory of your Linux VMs and OSS databases, you can start to determine the most appropriate strategy for reducing complexity and maximising ROI as you migrate and modernise.

There are many versions of cloud rationalisation (the 'R's), but for open-source-heavy environments, use these seven 'R's to rationalise your portfolios:

- **Rehost** to lift-and-shift to VMs.
- **Replatform** to move to managed services (PaaS) without major code changes.
- **Refactor** to make small code changes without changing what an app does.
- **Rearchitect** to redesign your core architecture using cloud-native capabilities.
- **Replace** to move to Software-as-a-Service (SaaS).
- **Retire** to decommission.
- **Retain** to keep as-is for now.

Rationalisation rule of thumb

When rationalising workloads, start with pragmatic steps that deliver value quickly. The following options reflect common business and technical goals.

Goal:
Remove distribution sprawl to reduce operational overhead and compliance risk.

Path:
Rehost to standardise

- Easier to secure.
- Automated patching and monitoring.
- Less sprawl makes it easier to modernise later.

Goal:
Reduce ops complexity caused by OSS databases or middleware.

Path:
Replatform

- Automated backup and patching
- Built-in replication and failover.
- Dynamic scalability with predictable SLAs.

Goal:
App portability and scale without fully rearchitecting.

Path:
Refactor using containers

- Modern deployment model.
- Path to microservices, DevOps and cloud-native architectures.
- Less OS management overhead.
- Portability across clouds.

Goal:
End-to-end modernisation.

Path:
Rearchitect using microservices

- Lower long-term cost from added elasticity.
- Ideal for workloads with variable demand.
- Agility to update, scale and deploy services independently.
- Faster time to market – release features without full app updates.

Considerations

- ### Don’t lift everything, then figure it out.

  Through rationalising, you can address urgent needs with an eye toward long-term value – rehost now to stabilise workloads, replatform databases to unify your data estate, containerise soon to offload more overhead and rearchitect selectively as time and budget allow.

- ### Do start small.

  To build momentum, identify a few workloads to replatform, such as applications that share databases. You can minimise risk while gaining experience, test your target architecture and validate your approach with a proof of concept.

- ### Prepare OSS data upfront.

  With minimal code changes, you can move OSS databases to cloud-managed database services, bringing apps and data together to improve performance and cost-efficiency while reducing latency. You can continue to build with your favourite extensions, frameworks and languages with support for the latest community versions.

- ### Skill up.

  Make sure your teams have the necessary Linux and OSS expertise to manage, migrate and operate workloads effectively in the target environments. suppliers that offer role-specific training and personalised learning plans can help your teams get up to speed quickly.

- ### Decide on governance.

  Ensure your landing zones define policies and controls for deploying, securing and managing Linux and OSS resources. Each rationalisation option introduces specific requirements – such as compliance standards, patching policies, identity and access management and cost controls – that your governance model should address.

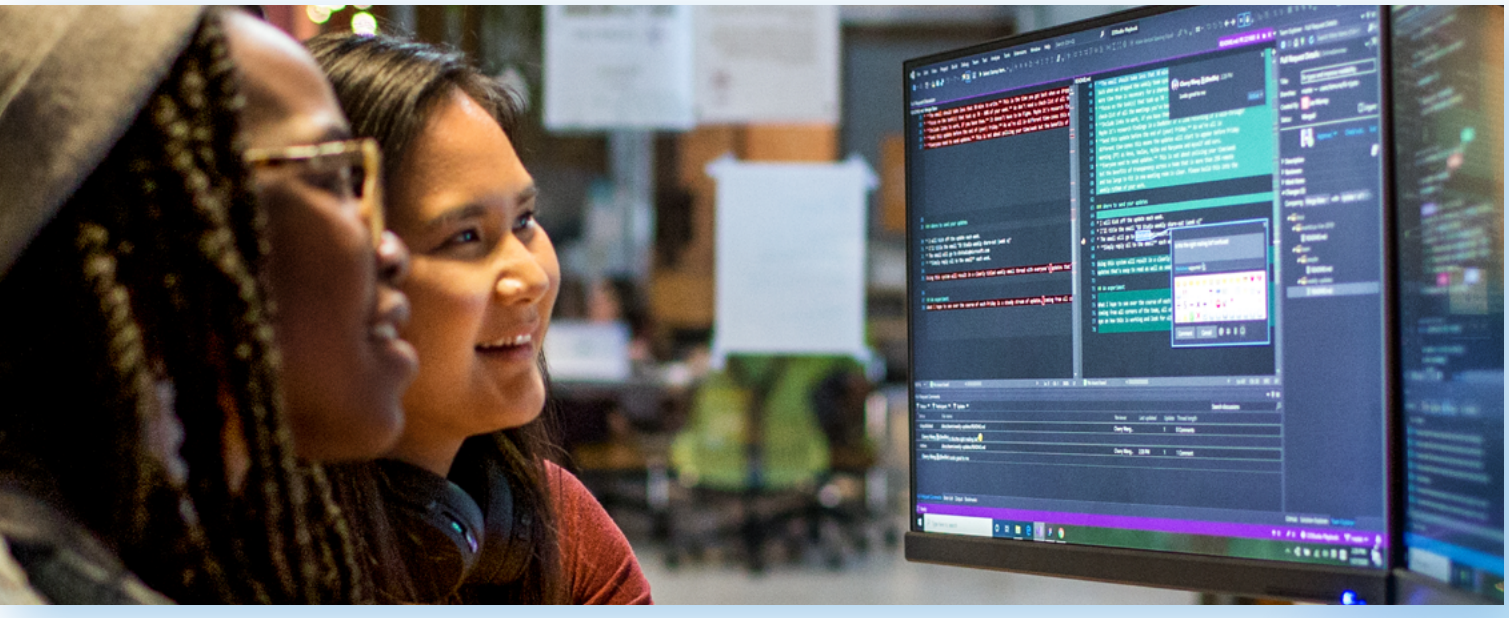
Rehost to standardise and stabilise

Rehosting enables you to quickly move workloads with minimal disruption, improving business continuity for on-premises databases and stabilising systems facing end-of-support or urgent needs. It offers a cost-effective path for legacy workloads not yet ready for full modernisation. For example, lifting and shifting to a Linux VM on Azure consolidates fragmented environments under a single, supported platform – reducing operational complexity and easing future upgrades.

Rehosting early in the process also surfaces hidden dependencies – like cross-schema joins or OS-bound cron jobs – helping you make informed decisions about which workloads to replatform, refactor or retire.

When to choose rehosting

- You manage multiple Linux distributions and apps are tightly coupled to the OS.
- Compliance or security mandates a standardised baseline across environments.
- You have a short migration window, and containerisation isn't practical.
- You need full database control, and platform services aren't a fit.



How to choose a distribution

As you review your current Linux mix, you can replace unsupported or outdated distributions and standardise on a supported release with better security and long-term support. It helps to define your licensing model upfront so you can avoid surprises and optimise costs. Flexible pricing and licensing options include pay-as-you-go for variable demand, bring your own subscription to maximise existing supplier agreements or reserve capacity for predictable workloads to optimise long-term costs. You can choose from ready-to-run VM and container images for applications, databases, middleware, developer tools and security solutions – validated for compatibility and security within the Microsoft cloud ecosystem. You can even use a base image from Azure Marketplace, customise it using Azure Image Builder and store that image in your own private image gallery.

Microsoft works with the community to ensure that most major distributions and many smaller ones run smoothly on Azure. Our policy is simple: We welcome all varieties of Linux. If you aren't tied to a specific distribution, we recommend starting with our list of endorsed distributions. Today that includes Red Hat Enterprise Linux (RHEL), Ubuntu, SUSE Linux Enterprise Server (SLES), Debian, Flatcar, Oracle Linux and the latest – Rocky Linux and AlmaLinux. Endorsed doesn't mean recommended, but rather shows that Microsoft and the distribution provider have a formal agreement to ensure a regular update cadence and remediation targets for security and stability. Endorsed distributions are in demand in the market and widely in use on Azure. In addition, Microsoft and the provider establish an engineering relationship and collaborate on testing and integration. You get predictable updates and integrated support from trusted suppliers like Red Hat, Canonical and SUSE, so your systems remain stable without extra effort. In addition, you can apply Azure Hybrid Benefit for Linux to existing RHEL or SLES subscriptions, when eligible, to optimise cost and maintain supplier support.

Azure and open source

Leading open-source partners like Red Hat, Canonical, SUSE and the Cloud Native Computing Foundation (CNCF) community work with Microsoft to deliver enterprise-grade Linux solutions optimised for Azure. These partnerships power the three most widely used Linux distributions:

→ RHEL

RHEL on Azure provides the security, compliance and joint Microsoft-Red Hat support trusted by 90% of Fortune 500 companies. It's ideal mission-critical workloads like SAP HANA, SQL Server and Java, with managed services for Ansible, OpenShift and JBoss, plus automation for SAP and high-performance computing environments.

→ Ubuntu

Ubuntu is the most popular Linux distribution on Azure, backed by a deep Microsoft-Canonical engineering partnership. It offers optimised images, five years of LTS security updates and unique features like Confidential GPU VM support for secure AI and data processing.

→ SUSE

SUSE leads in SAP environments, with 70% of SAP applications and 85% of SAP HANA systems running on SUSE Linux. Microsoft and SUSE co-develop solutions like Rancher Prime and Azure Arc Jump-start accelerators for hybrid and multicloud management, delivering agility, security and simplified operations.

Replatform databases to unify and simplify

Database migration unlocks unified data – a strategic move from fragmented systems to a single, secure, cloud-native platform for apps and data. You can replace data silos and fragile integrations with a governed environment that provides high-quality, trusted data.

By lowering the integration overhead, your IT teams gain a foundation for real-time analytics, AI-powered applications and scalable innovation. Even a 10% boost in data usability can translate into USD 2 billion more annual revenue for the average Fortune 1000 company.[1]

Replatforming databases and apps is a key step toward scalability and lower TCO. Unlike lift-and-shift approaches that retain OS-level dependencies and overhead, managed services abstract the OS and kernel and provide automated scaling, patching and compliance. You get predictable performance for your workloads and less operational overhead for your teams – both critical for microservices architectures that demand agility and reliability.

Replatforming also makes sense if you plan to use containers and microservices. By using managed service endpoints, your microservices can communicate more reliably and securely, reducing configuration overhead and improving architectural resilience. If you design your landing zone to include a managed database service alongside your container platform from the start, you don't have to manage the database yourself within the cluster or container.

Considerations

- **Minimise downtime.** For less critical workloads, a simple downtime migration can be the fastest path – using tools like Azure Data Box for large offline transfers. For business-critical systems, near-zero downtime is essential. Continuous replication and planned cutover, supported by services such as Azure Database Migration Service, help you minimise disruption and keep operations running smoothly.
- **Build connectivity in.** Landing zones help you prepare the network, identity and security foundation so any database modernisation path – VM-hosted, containerised or PaaS – is supported without rework. You can ensure secure, reliable and performant access for Linux apps and OSS databases.
- **Enable AI-driven innovation.** Modernising your databases on cloud-native, open-source platforms creates a foundation for AI workloads without adding complexity. For example, Azure Database for PostgreSQL supports vector search and retrieval-augmented generation (RAG), so you can build intelligent applications using familiar tools and extensions – without standing up a separate data platform.
- **Quick Oracle migration.** With minimal refactoring, you can move to Oracle Database@Azure and modernise with integrated Microsoft services. This approach provides a broad range of Oracle database capabilities along with the regional availability and resiliency of Azure.

[1] CIO vision 2025: Bridging the gap between BI and AI. Databricks. MIT Technology Review Insights. 2025.

Refactor using containers as the bridge to cloud-native

Refactoring into containers is a strategic middle ground – you can modernise your deployment model without the cost of fully rearchitecting. Containers help you avoid dependency conflicts across multiple Linux distributions. They ensure that applications run consistently across development, test and production environments. By decoupling apps from the OS lifecycle, containers give you portability, flexibility and faster delivery.

Containerisation abstracts the OS layer, so apps run in a consistent environment regardless of the underlying host distribution. Instead of managing five distributions across 200 VMs, you can:

- → **Move apps into containers with a base image you control.**
- → **Use managed services to run them consistently.**

Containers are the foundation for microservices, DevOps and cloud-native architectures. For example, if you have a Linux app running on a VM, you can package it as-is into a container image hosted on a managed service like Azure Kubernetes Service (AKS). Without changing the app's internal structure, you can add elasticity, resilience and horizontal scaling while integrating your app with modern observability and security tools. The move also sets you up to make improvements later – such as replatforming databases or containerising more apps – without redoing identity and governance.

Considerations

- **Provide AKS-ready networking.** Kubernetes clusters have unique requirements for connectivity, security and scalability. You can set up a landing zone that prepares the network foundation so AKS clusters can be deployed securely and at scale without rearchitecting later.
- **Keep your ecosystem.** If you standardise on Ubuntu, SLES or RHEL, you can run containers on Azure-endorsed images to maintain OS consistency and existing tooling. If your organisation is already invested in OpenShift, Azure Red Hat OpenShift (ARO) provides a managed, enterprise OpenShift environment with built-in continuous integration and continuous delivery (CI/CD), security and policy controls – ideal if you want to keep your OpenShift skills and workflows while offloading cluster management.
- **Evaluate database containerisation strategically.** Containerising databases can introduce complexity, because databases are stateful while containers are designed to be ephemeral. Unless you have a strong business or technical driver – such as portability across environments or custom requirements that make PaaS impractical – replatforming to a managed service is often the better choice. If you do containerise, ensure persistent storage and failover are properly managed to avoid data integrity risks.

Rearchitect for scale with microservices

Rearchitecting legacy Linux applications and databases into a microservices framework is one of the most effective ways to accelerate innovation and scale.

Instead of deploying an entire monolithic application, teams can update and scale individual services – like the cart in a shopping app – on their own. This flexibility speeds up feature delivery, improves resilience (a single service failure won't take down the whole system) and drives meaningful cost optimisation. Plus, it positions your organisation squarely in line with modern cloud-native practices.

Success starts with a strong data foundation. Establishing a solid data foundation ensures each service has an optimised, AI-ready data store and consistent access, keeping your architecture aligned with cloud-native best practices from the start.

Considerations

- **Choose your orchestrator upfront.** Your orchestration platform – whether AKS, ARO or Arc-enabled Kubernetes – affects compliance, automation and integration with your Linux ecosystem. Decide early to align governance and operational models.
- **Build your way.** A code-to-cloud toolchain simplifies Kubernetes management and gives teams the flexibility to adopt the best-fit language, framework or OSS component for each service – avoiding supplier lock-in and extending the life of Linux-based investments.
- **Align your architecture and landing zone.** To fit microservices into a well-governed strategy, align DevOps pipelines with landing zone designs and use cloud-native role-based access control, policies and network segmentation.
- **Plan for Zero Trust from the start.** Modernisation is an opportunity to embed Zero Trust principles into your architecture. Define identity-based access, enforce least privilege for service-to-service communication, and ensure network segmentation is part of your design – so security scales with your cloud-native environment.
- **Bake security into your container strategy.** Modernisation is the right time to embed security into your container lifecycle. Plan for image scanning, compliance checks and runtime protection as part of your orchestration and CI/CD approach – using tools like Microsoft Defender for Cloud to ensure security scales with your cloud-native environment.

Unify governance for consistent outcomes

A unified approach to governance ensures that every workload – whether rehosted, replatformed or refactored – operates within your organisation's standards without adding operational complexity. Governance turns migration into a repeatable factory. Consistent security baselines cut down rework, policy-driven compliance reduces audit risk and cost guardrails prevent overruns as you scale Linux and OSS estates.

If your cloud rationalisation strategy includes keeping some workloads on-premises, plan for a hybrid environment from the start. Hybrid impacts landing zone design, controls and the sequencing of modernisation efforts. With consistent policy enforcement across on-premises and cloud – covering identity, patching and compliance – you can simplify operations and ensure smooth migrations. Because hybrid is often a transitional state, design governance to adapt as workloads move to the cloud, avoiding costly re-engineering later.

Considerations

- **Start small, scale fast.** To accelerate migrations, start with a small, actionable set of standards – identity and access, image and patch baselines, data and network protections, tagging and budgets – and expand as your environment grows. You can apply standards uniformly across VMs, containers and managed services as you go from rehosting to refactoring and into full cloud-native transformation.
- **Unify and automate.** Fragmented visibility creates gaps in security and governance, while manual processes increase the risk of drift. A unified governance and security framework reduces complexity and improves resilience. For example, you can use Azure Policy to enforce configurations, Azure Arc to extend governance across hybrid environments and Microsoft Defender for Cloud to strengthen security posture – all with automated compliance and monitoring.
- **Automate guardrails.** Use policy-driven automation to enforce standards consistently across workloads. You can reduce manual effort and ensure compliance without adding friction.
- **Align with business priorities.** Governance should enable – not block – business goals. Focus on controls that support security, compliance and cost efficiency while allowing teams to move quickly.
- **Enforce when ready.** Start with visibility and guidance, then move to enforcement as your governance model matures. This phased approach helps teams adopt standards without slowing migration, while ensuring compliance, security and cost controls scale with your environment and workload growth.

Operate, optimise and evolve: the path to lasting value

While earlier steps focus on enabling migration and modernisation, operational consistency is what transforms those efforts into sustained business value – through efficiency, security and the agility to innovate at scale.

It's not just about running workloads; it's about ensuring they remain secure, compliant and cost-optimised as your environment evolves.

Considerations

- **Commit to continuous improvement.** Modernisation isn't a finish line – it's an ongoing journey. Use automation, consistent security baselines and real-time telemetry to optimise performance, scale and cost. Regularly revisit architecture and KPIs to ensure your environment evolves as fast as your business, laying the foundation for DevOps and FinOps maturity.
- **Design for resilience, not just uptime.** Think beyond availability – build systems that adapt to change, recover gracefully and evolve with your business needs.
- **Automate everything you can.** Automation isn't just efficiency; it's consistency at scale. Every manual process you eliminate reduces risk and accelerates innovation.
- **Measure what matters.** Define KPIs that align with business outcomes like cost efficiency, security posture and deployment velocity, not just technical metrics.
- **Embed security and compliance.** Treat security as a design principle, not an afterthought. Continuous compliance monitoring keeps you ahead of risk and regulation.
- **Empower teams with observability.** Give teams real-time insights into performance, cost and compliance so they can act quickly and confidently.

Next steps

Get end-to-end guidance, find experts and unlock funding

Explore Azure Accelerate

Learn about available offers

Contact an Azure sales specialist