THE OPENSTACK HANDBOOK

FROM FIRST DEPLOYMENT TO PRODUCTION-SCALE CLOUD OPERATIONS

PLAN
ARCHITECT AND DESIGN YOUR CLOUD

DEPLOY
BUILD YOUR CLOUD WITH CONFIDENCE

CONFIGURE
FINE-TUNE SERVICES FOR RELIABILITY AND PERFORMANCE

SECURE
PROTECT YOUR CLOUD AND YOUR DATA

OPERATE
MANAGE AND OPTIMIZE DAILY OPERATIONS

TROUBLESHOOT
DIAGNOSE ISSUES AND KEEP YOUR CLOUD HEALTHY

RYAN HOLLOWAY

The OpenStack Handbook

From First Deployment to Production-Scale Cloud Operations

Steve Publications

This book is available at https://leanpub.com/theopenstackhandbook

This version was published on 2026-08-12

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Contents

From First Deployment to Production-Scale Cloud Operations

This book is a comprehensive, practical guide to planning, deploying, configuring, administering, securing, operating, and troubleshooting OpenStack clouds. Written for system administrators, DevOps engineers, and cloud architects, it takes you from fundamental concepts through production-grade operations with detailed procedures, realistic configuration examples, and systematic troubleshooting techniques. Whether you are building your first lab environment or managing a multi-thousand-node enterprise deployment, this book serves as both a learning resource and an ongoing reference manual.

Introduction: Why OpenStack Still Matters

The State of Cloud Computing in 2026

The cloud computing landscape has matured dramatically over the past decade. Public cloud providers like Amazon Web Services, Microsoft Azure, and Google Cloud Platform dominate general-purpose workloads, offering virtually unlimited scale and an ever-expanding catalog of managed services. For many organizations, especially startups and small to medium enterprises, public clouds are indeed the right choice.

Yet a significant portion of the world's most demanding workloads continues to run on private and hybrid cloud platforms built with OpenStack. Major telecommunications operators, financial institutions, government agencies, research organizations, and hyperscale providers all rely on OpenStack for their infrastructure-as-a-service needs. Why? Because there are scenarios where public clouds do not fit: data sovereignty requirements that mandate on-premises processing, latency-sensitive applications that cannot tolerate the round trip to a distant cloud region, cost structures where predictable capital expenditure beats variable operational spend at scale, and regulatory frameworks that require complete control over the infrastructure stack.

OpenStack addresses these needs as an open source IaaS platform that gives organizations the same fundamental capabilities found in public clouds: self-service provisioning of virtual machines, software-defined networking, elastic block storage, object storage, orchestration, and load balancing. The key difference is control. With OpenStack, you own the stack. You decide where data lives, how networks are segmented, what security policies are enforced, and how resources are allocated.

The platform's relevance has not diminished with time. As of 2026, OpenStack releases continue on a six-month cadence, with the current stable release being 2026.1, codenamed Gazpacho. The community introduces Skip

Level Upgrade Release Process releases that allow operators to upgrade annually rather than every six months, reducing operational burden while maintaining access to security fixes and improvements. Major vendors including Canonical, Red Hat, SUSE, and Mirantis continue to invest heavily in OpenStack distributions, ensuring enterprise-grade support and integration with complementary technologies like Kubernetes, Ceph, and container orchestration platforms.

What OpenStack Is and Is Not

OpenStack is a collection of interrelated projects that together provide an IaaS platform. Each project implements a specific service: compute provisioning, networking, storage, identity management, image handling, orchestration, and more. These services communicate through well-defined APIs, allowing them to be deployed independently or in combination depending on your needs.

OpenStack is not a monolithic application. It is modular by design. You might deploy only the compute and networking services without block storage if that meets your requirements. You might add object storage later when your use case demands it. This modularity is both OpenStack's greatest strength and its primary source of complexity. Understanding how the pieces fit together is essential to successfully deploying and operating an OpenStack cloud.

OpenStack is also not a turnkey solution in the traditional sense. While deployment tools exist that automate much of the installation process, running OpenStack successfully requires genuine expertise in Linux administration, networking, storage systems, virtualization, and distributed systems concepts. There is no single configuration that works for every environment. Each deployment must be designed around specific hardware, network topology, workload characteristics, and operational requirements.

OpenStack does not replace public clouds; it complements them. Many organizations use OpenStack for their private cloud while leveraging public clouds for burst capacity, specialized services, or development environments. Tools like Terraform and the OpenStack CLI enable consistent management across both environments through infrastructure-as-code practices.

Who Uses OpenStack and Why

The OpenStack user base spans a wide range of organizations with diverse requirements:

Telecommunications operators use OpenStack extensively for network functions virtualization, deploying virtualized network functions on standardized hardware rather than proprietary appliances. The platform's networking capabilities, combined with support for bare metal provisioning through the Ironic service, make it well suited for carrier-grade deployments that require high availability and strict performance guarantees.

Financial institutions deploy OpenStack to meet regulatory requirements while gaining cloud-like agility for development and testing environments. Complete control over data placement, encryption, and access policies satisfies compliance mandates that would be difficult or impossible to achieve with public cloud providers alone.

Government agencies worldwide rely on OpenStack for sovereign cloud deployments. National governments in Europe, Asia, and the Americas have built large-scale OpenStack clouds to provide secure, compliant infrastructure for public sector workloads while maintaining data residency within national borders.

Research and education institutions use OpenStack to provide shared computing resources across departments and campuses. The platform's multi-tenant architecture with project-based resource allocation and quota enforcement makes it ideal for environments where multiple groups need isolated access to shared infrastructure.

Large technology companies and cloud providers themselves sometimes build their own public clouds on OpenStack, customizing the platform to meet specific performance, scale, or feature requirements that differ from mainstream public cloud offerings.

How This Book Is Organized

This book is structured to take you on a progressive journey from foundational understanding through production operations. Each chapter builds on previous material, so reading in order is recommended for those new to OpenStack.

Experienced operators can use it as a reference, jumping directly to chapters relevant to their immediate needs.

The Introduction provides this context about OpenStack's role in modern cloud computing and sets expectations for the book. Chapter 1 establishes the architectural foundation by explaining each major service, its purpose, and how services interact with one another. You cannot effectively deploy or troubleshoot OpenStack without understanding this architecture.

Chapters 2 and 3 cover planning and preparation. Chapter 2 addresses deployment scenarios from small labs to enterprise-scale clouds, hardware sizing, network design, storage planning, and high availability considerations. Chapter 3 walks through base system preparation: installing the operating system, configuring networking, setting up DNS and time synchronization, and preparing package repositories.

Chapters 4 through 9 detail the installation and configuration of each major service. Chapter 4 covers core infrastructure services that everything else depends on: database, message queue, caching, and identity. Chapter 5 installs compute services. Chapter 6 covers image management. Chapter 7 tackles networking, the most complex service in OpenStack. Chapters 8 and 9 cover block storage, object storage, and Ceph integration.

Chapter 10 covers additional services: the Horizon dashboard, Heat orchestration, Octavia load balancing, Magnum containers, and Manila file sharing. Chapter 11 focuses on daily operations: creating resources, managing users and projects, launching instances, handling quotas, and performing routine administrative tasks.

Chapter 12 addresses security comprehensively: identity hardening, role-based access control, TLS encryption, certificate management, network isolation, firewalling, and compliance considerations. Chapter 13 covers automation through APIs, the OpenStack CLI, client SDKs, Terraform, Ansible, and CI/CD integration.

Chapter 14 brings together monitoring, performance optimization, backup and disaster recovery, upgrade procedures, and systematic troubleshooting methodologies. The Conclusion synthesizes lessons from production deployments and offers guidance for long-term success operating OpenStack at scale.

Throughout the book, configuration examples use realistic values rather than generic placeholders. Commands are shown in context with explanations of what they do and why each step is necessary. Where multiple approaches

exist, trade-offs are discussed so you can make informed decisions for your specific environment.

Chapter 1: OpenStack Architecture and Core Concepts

The Big Picture: Controllers, Compute Nodes, Network Nodes, Storage Nodes

OpenStack deployments consist of physical or virtual servers organized into logical roles based on the services they run. Understanding these roles is essential before diving into individual services.

The controller node runs the management and API components of OpenStack services. It hosts the database where all configuration data and state information is stored, the message queue that enables service-to-service communication, the identity service for authentication, and the API endpoints that clients use to interact with the cloud. In production deployments, you typically run multiple controller nodes in a highly available cluster to eliminate single points of failure.

The compute node runs virtual machines using a hypervisor such as KVM, VMware ESXi, or XenServer. The nova-compute service on each compute node communicates with the controller to receive instructions about which instances to create, modify, or destroy. Compute nodes are where actual workload processing happens and are scaled horizontally based on demand.

The network node handles networking functions that cannot be distributed to compute nodes: routing between networks, NAT for external connectivity, DHCP services for tenant networks, and sometimes load balancing. In modern deployments using OVN or distributed virtual routing, many of these functions can be spread across compute nodes, reducing the need for dedicated network nodes in some scenarios.

The storage node provides block storage volumes through Cinder or object storage through Swift. When using Ceph as a unified storage backend, storage nodes run Ceph OSD daemons that distribute and replicate data across the cluster. Storage nodes may also host Glance image storage backends.

Many deployments use converged nodes that combine roles. A common pattern is to co-locate controller services on separate dedicated servers while running compute, network, and storage functions together on converged infrastructure nodes. This reduces hardware requirements at the cost of increased complexity in capacity planning and resource isolation.

The communication between these nodes relies on several distinct networks:

- The management network carries traffic between OpenStack services themselves, including API calls between components, database connections, and message queue traffic. This network should be isolated from external access.
- The API or tenant network provides external access to the OpenStack API endpoints and Horizon dashboard. This is where users and automation tools connect to manage the cloud.
- The storage network handles traffic between compute nodes and storage backends, including Ceph cluster communication, iSCSI connections for Cinder volumes, and Swift object replication. Dedicated high-speed networking (10 Gbps or higher) is recommended here.
- The provider or external network connects virtual machines to external networks such as the internet or corporate LANs through floating IPs and routers.

Keystone: Identity and Authentication

Keystone is the foundation upon which all other OpenStack services are built. Every service authenticates through Keystone, every user is authenticated through Keystone, and every API request must present a valid token issued by Keystone. Without a functioning Keystone deployment, no other OpenStack service can operate.

Keystone implements four core functions:

- Identity management: storing users, groups, projects, and domains
- Authentication: verifying credentials and issuing tokens
- Authorization: determining what actions authenticated entities are permitted to perform through policy files

- Service catalog: maintaining a registry of all available services and their API endpoints

Users in OpenStack belong to projects, which provide resource isolation and quota enforcement. A project is analogous to a tenant or account in commercial cloud platforms. Multiple users can belong to the same project and share its resources, subject to role-based permissions. Domains provide an additional layer of organization for large deployments with multiple organizations or business units, each maintaining their own user directory.

Keystone supports multiple authentication methods beyond simple username and password: LDAP integration for existing directory services, SAML federation for single sign-on, multi-factor authentication through plugins, and token-based authentication for service-to-service communication. The default token format in modern deployments is Fernet, which uses symmetric encryption to create self-contained tokens that do not require database lookups for validation, improving performance under load.

When a user authenticates, Keystone issues a token containing the user's identity, project membership, and roles. This token is included with every subsequent API request. Services validate the token locally using Fernet keys shared between them, eliminating the need to query Keystone for every request while maintaining security through cryptographic validation.

The service catalog maintained by Keystone is critical for inter-service communication. When Nova needs to communicate with Glance to retrieve an image, it queries the service catalog to find the correct Glance API endpoint rather than having a hardcoded address. This enables flexible deployment topologies where services can be moved or scaled without reconfiguring every dependent component.

Nova: Compute Orchestration

Nova is the compute service that manages the lifecycle of virtual machine instances. It does not run hypervisors directly but orchestrates them through drivers, with KVM being the most common choice in production deployments. Nova handles everything from instance creation and scheduling to live migration, resizing, and termination.

Nova's architecture is distributed across several components:

- nova-api receives REST API requests from users and other services, validates them against policies, and queues operations for processing
- nova-scheduler selects the optimal compute node for new instances based on available resources, configured filters, and weight functions
- nova-conductor acts as a secure intermediary between other Nova components and the database, preventing direct database access from compute nodes
- nova-compute runs on each compute node, managing local instances through the hypervisor and reporting resource usage back to the scheduler
- nova-novncproxy provides VNC console access for instance troubleshooting

When you request a new instance, the process flows through these components systematically. The API receives your request specifying image, flavor, network, and other parameters. It validates that you have sufficient quota and permissions, then places a scheduling request on the message queue. The scheduler evaluates all available compute nodes using filter drivers that eliminate unsuitable candidates based on criteria like available memory, disk space, CPU architecture, and hardware capabilities. Remaining candidates are scored by weight functions that prefer nodes with more free resources or specific characteristics. The scheduler selects the highest-scoring node and returns its identity to the API.

The API then instructs the chosen compute node to build the instance. The nova-compute service on that node retrieves the image from Glance, allocates network ports through Neutron, provisions any requested volumes from Cinder, and creates the virtual machine using the hypervisor. Throughout this process, resource allocations are tracked in the Placement service to maintain accurate inventory of available capacity across the cloud.

Nova supports multiple hypervisors through its driver architecture: KVM for Linux environments, VMware ESXi for existing VMware deployments, XenServer, Hyper-V, and even container-based execution through Docker integration. This flexibility allows gradual migration between virtualization platforms or coexistence of different hypervisors within a single OpenStack deployment.

Neutron: Networking as a Service

Neutron provides software-defined networking capabilities that allow users to create and manage their own virtual networks within the cloud. It is arguably the most complex OpenStack service due to the breadth of networking functions it must support and the variety of underlying technologies it can leverage.

The core concept in Neutron is the Modular Layer 2 plugin, or ML2, which separates network type drivers from mechanism drivers. Type drivers define how networks are realized: flat networks that map directly to physical networks, VLAN-tagged networks for provider segmentation, VXLAN or Geneve overlay networks for tenant isolation, and GRE tunnels as an alternative overlay protocol. Mechanism drivers define how these networks are implemented on the infrastructure: Open vSwitch, Linux bridge, OVN (Open Virtual Network), SR-IOV for direct hardware access, and vendor-specific plugins.

This modular architecture allows a single deployment to support multiple networking modes simultaneously. Provider networks might use VLANs for direct physical network integration while tenant networks use VXLAN overlays for multi-tenancy. High-performance workloads can leverage SR-IOV for near-bare-metal network performance while general-purpose instances use standard virtual interfaces.

Neutron manages several key networking objects:

- Networks represent Layer 2 broadcast domains, analogous to physical switches or VLANs
- Subnets define IP address ranges and associated configuration within a network
- Ports are attachment points on networks where instance interfaces connect, each with its own MAC address and optional IP addresses
- Routers provide Layer 3 connectivity between networks and NAT for external access
- Security groups act as stateful firewalls controlling traffic to and from instances

Provider networks are administrator-managed networks that typically connect directly to physical infrastructure. They are used for external connectivity, management access, or dedicated tenant networks where direct VLAN

assignment is required. Self-service or tenant networks are created by users within their projects and are isolated from other tenants through overlay protocols or VLAN segmentation.

The Neutron architecture distributes functions across several agents:

- The L2 agent (Open vSwitch, Linux bridge, or OVN) manages virtual switches and network interfaces on compute and network nodes
- The L3 agent handles routing between networks and NAT operations
- The DHCP agent provides IP address assignment to instances within tenant networks
- The metadata agent serves instance metadata requested by cloud-init during boot
- Additional agents support load balancing, QoS enforcement, metering, and other extensions

Floating IPs are a critical feature for external connectivity. They map public IP addresses to the private IPs of instances on internal networks, enabling inbound access without exposing the internal network topology. Floating IPs are allocated from a pool associated with an external provider network and can be dynamically attached and detached from instances.

Glance: Image Management

Glance is the image service that provides discovery, registration, and delivery of virtual machine disk images. It serves as the central repository for all bootable images in the cloud, supporting multiple storage backends and image formats.

Glance does not create images; it stores, retrieves, and manages them. Images are typically created outside OpenStack using tools like virt-builder, Packer, or cloud-image utilities that produce optimized disk images for specific distributions. These images are then uploaded to Glance where they become available for launching new instances.

Supported image formats include raw (uncompressed disk images), QCOW2 (QEMU copy-on-write format with support for snapshots and compression), VMDK (VMware format), VHD (Hyper-V format), and others. The choice of format depends on your storage backend and performance requirements. Raw

images generally offer better performance with Ceph RBD backends, while QCOW2 provides space efficiency through thin provisioning when using file-based storage.

Glance supports multiple storage backends through its store architecture:

- File system storage uses local disk space, suitable for simple deployments but lacking redundancy
- Swift object storage provides distributed, fault-tolerant storage for images
- Ceph RBD offers high-performance block storage with thin provisioning and cloning capabilities
- HTTP stores enable proxying to external image locations without storing full copies

When an instance is launched, Nova retrieves the boot image from Glance. With Ceph backends, this process leverages copy-on-write cloning rather than copying entire images, enabling near-instantaneous instance creation even for large images. This efficiency is one of the primary reasons production deployments typically integrate Glance with Ceph.

Glance also manages image metadata including minimum RAM and disk requirements, architecture specifications, hypervisor type, and custom properties that can be used for filtering and selection. Image visibility can be set to public (available to all projects), private (visible only within the project that created it), or shared (explicitly shared with specific projects).

Cinder: Block Storage

Cinder provides persistent block storage volumes that can be attached to running instances or used as boot devices. Unlike ephemeral instance storage that is deleted when an instance is terminated, Cinder volumes persist independently and retain their data across instance lifecycle events.

Cinder's architecture separates API handling from volume management. The cinder-api service processes requests for creating, attaching, detaching, and deleting volumes. The cinder-scheduler selects the appropriate backend for new volumes based on available capacity, performance characteristics, and configured policies. The cinder-volume service runs on storage nodes and

manages actual volume creation and lifecycle operations on the configured backend.

Multiple backends can be configured within a single Cinder deployment through volume types. Each volume type maps to specific backend configurations and capabilities. For example, you might define an ssd volume type backed by NVMe drives for high-performance workloads and a standard volume type using SATA drives for cost-sensitive storage. Users select volume types when creating volumes, enabling differentiated service levels within the same cloud.

Supported backends include:

- LVM with iSCSI for simple deployments using local storage
- Ceph RBD for distributed, highly available block storage (most common in production)
- NFS for shared file system-backed storage
- Vendor-specific drivers for enterprise SAN systems from NetApp, Dell EMC, HPE, and others

Cinder volumes support snapshots that capture point-in-time copies without disrupting running instances. Snapshots can be used for backup purposes or to create new volumes with existing data. Volume cloning creates independent copies of volumes, useful for testing or development scenarios. Cinder also supports volume replication between backends for disaster recovery and live migration between storage systems.

When an instance boots from a Cinder volume rather than ephemeral storage, the volume serves as the root disk. This enables persistent root filesystems that survive instance termination (when configured appropriately) and simplifies backup and recovery procedures since all data resides on manageable block devices.

Swift: Object Storage

Swift provides distributed object storage for unstructured data such as backups, media files, logs, and archives. Unlike block storage which presents fixed-size devices, object storage manages individual objects identified by unique keys within containers (analogous to buckets in other cloud platforms).

Swift's architecture is designed for massive scale and fault tolerance without relying on complex consensus protocols. It distributes data across multiple nodes using a consistent hashing algorithm implemented through ring files that map objects to physical storage locations. Each object is replicated across multiple nodes, with the replication factor configurable at deployment time (typically three replicas for production).

The Swift stack consists of several components:

- The proxy server handles all client requests, authentication, routing, and metadata operations
- Account servers track container listings for each account
- Container servers manage object listings within containers
- Object servers store the actual data on disk
- The ring builder utility calculates data distribution across the cluster

Swift's simplicity is intentional. There is no centralized metadata database that could become a bottleneck or single point of failure. Instead, all metadata is distributed alongside the data itself, enabling linear scalability as nodes are added to the cluster. This design makes Swift particularly well suited for large-scale deployments storing petabytes of data.

In OpenStack environments, Swift serves multiple purposes beyond user-facing object storage: it can back Glance image storage, serve as a backup target for Cinder volumes, store telemetry data from monitoring services, and provide general-purpose object storage for applications running in the cloud. While Ceph has become more popular for unified storage in many deployments, Swift remains relevant for specialized use cases requiring massive scale or when organizations already have Swift expertise.

Horizon: The Web Dashboard

Horizon provides a web-based graphical interface to OpenStack services, enabling users and administrators to manage cloud resources through a browser instead of command-line tools. Built on the Django framework, Horizon translates user actions into API calls against the underlying OpenStack services.

Horizon organizes its interface around three primary perspectives:

- The admin panel provides cloud administrators with global management capabilities: managing users, projects, domains, network infrastructure, and system-wide settings
- The project panel enables regular users to manage resources within their assigned projects: launching instances, creating networks, managing volumes, uploading images, and monitoring usage
- The setting panel allows users to configure personal preferences, API access credentials, and notification settings

Horizon is modular through its plugin architecture. Additional panels can be added to support services like Heat orchestration, Octavia load balancing, Magnum containers, or third-party integrations without modifying the core dashboard. This extensibility ensures Horizon remains relevant as new services are added to OpenStack deployments.

While Horizon is invaluable for interactive management and exploration of the cloud environment, production automation typically relies on CLI tools, APIs, or infrastructure-as-code platforms rather than manual dashboard operations. Horizon excels at ad-hoc tasks, troubleshooting, visualization, and serving non-technical users who prefer graphical interfaces over command-line interaction.

Supporting Services: Placement, Heat, Octavia, Magnum, Manila

Beyond the core services discussed above, OpenStack includes several supporting projects that extend its capabilities:

Placement is a dedicated service for tracking resource provider inventories and allocations. Extracted from Nova in the Stein release, Placement maintains authoritative information about available resources across all compute nodes, including CPU, memory, disk, network ports, and custom resource classes. The Nova scheduler queries Placement to find suitable hosts for new instances, ensuring accurate capacity tracking even when multiple services consume resources on the same physical hardware.

Heat provides orchestration capabilities through declarative templates written in YAML or JSON using the Heat Orchestration Template format. Templates define composite cloud applications consisting of multiple

interconnected resources: networks, instances, load balancers, databases, and more. When a template is deployed as a stack, Heat creates all defined resources in the correct order, managing dependencies automatically. This enables reproducible infrastructure deployments and simplifies complex multi-component application provisioning.

Octavia delivers load balancing as a service through dedicated virtual machines called amphorae that run HAProxy or similar load balancing software. Users create load balancers with listeners (defining protocols and ports), pools (groups of backend instances), health monitors, and optional Layer 7 policies for advanced traffic routing. Octavia integrates with Nova to manage amphora instances and Neutron for network configuration, providing fully automated, scalable load balancing without requiring dedicated hardware appliances.

Magnum enables container orchestration by managing clusters of Kubernetes, Docker Swarm, or other container platforms as first-class OpenStack resources. When you create a Magnum cluster, it provisions the necessary compute instances, networks, and storage, then deploys the chosen container orchestration software. This allows organizations to run containerized workloads alongside traditional virtual machines within the same cloud infrastructure.

Manila provides shared file systems accessible by multiple instances simultaneously through standard protocols like NFS and CIFS/SMB. Unlike block storage which is typically attached to a single instance, file shares enable collaborative workloads, application clusters requiring shared state, and integration with existing applications designed for network-attached storage.

Additional services include Ironic for bare metal provisioning, Trove for database-as-a-service, Sahara for big data cluster management, Senlin for cluster management, Zaqar for messaging, and many others. Each service follows the same architectural patterns: REST APIs, Keystone authentication, message queue communication, and modular design that allows independent deployment and scaling.

How the Pieces Fit Together: A Request Lifecycle

Understanding how OpenStack services interact is best illustrated by following a typical request through the system. Consider what happens when a user launches a new virtual machine instance:

The user authenticates against Keystone using their credentials, receiving a token that proves their identity and project membership. This token is included with every subsequent API call.

The user submits an instance creation request to the Nova API specifying the desired image, flavor (resource allocation), network, key pair, and any additional options like attached volumes. Nova validates the request against policies and checks that the user has sufficient quota within their project.

Nova queries Placement to determine available resource providers that can accommodate the requested instance size. The scheduler evaluates candidates using configured filters and weights, selecting the optimal compute node.

Before building the instance, Nova retrieves image metadata from Glance to verify the image exists and meets minimum requirements. If booting from a volume, Cinder provisions a new volume and populates it with the image data (or creates a clone when using Ceph).

Neutron allocates network ports on the requested networks, assigns IP addresses from subnets via DHCP allocation, and configures any applicable security group rules. The compute node's networking agent prepares virtual interfaces for the instance.

The selected nova-compute service creates the virtual machine using the hypervisor, attaching the boot disk, configuring network interfaces, and injecting metadata through the Neutron metadata service. The instance boots, cloud-init reads its configuration from the metadata service, sets up SSH keys, applies user data scripts, and completes initialization.

Throughout this process, services communicate via the message queue for asynchronous operations and query each other's APIs as needed using tokens obtained from Keystone. Database state is updated at each step to maintain consistency, and events are published for monitoring systems to track progress and detect failures.

This choreography involves multiple services working in concert, each performing its specialized function while relying on others for supporting capabilities. When any single service fails or becomes unresponsive, the entire workflow can be affected. This interdependence underscores the importance of high availability at every layer: redundant controllers with clustered databases and message queues, load-balanced API endpoints, distributed storage backends, and geographically diverse compute capacity when disaster recovery is required.

Chapter 2: Planning Your OpenStack Deployment

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Deployment Scenarios: Lab, PoC, Small Production, Enterprise Scale

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Hardware Requirements and Sizing Guidelines

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Operating System Selection and Support Matrix

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Network Design: Management, API, Storage, Provider Networks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

DNS, NTP, and Time Synchronization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Storage Planning: Local vs Shared, Ceph Considerations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

High Availability Architecture Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Choosing a Deployment Method

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Chapter 3: Prerequisites and Base System Preparation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Installing the Operating System

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Configuring Static Networking

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Setting Up Hostnames and DNS Resolution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

NTP and Time Synchronization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Package Repositories and Updates

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Firewall Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

SSH Key Setup and Access Control

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Kernel Parameters and System Limits

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Chapter 4: Installing Core Infrastructure Services

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Database Server Installation and Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Message Queue (RabbitMQ) Setup

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Memcached for Caching

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Keystone Identity Service Installation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Creating Initial Users, Projects, and Roles

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Generating the Admin Credentials File

Chapter 5: Deploying Compute Services with Nova

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Placement Service Installation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Nova Controller Components

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Configuring a Compute Node

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Verifying Compute Service Operation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Understanding Nova Scheduling and Resource Tracking

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Chapter 6: Deploying Image Services with Glance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Glance Controller Installation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Configuring Storage Backends for Images

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Uploading and Managing Images

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Image Formats, Properties, and Best Practices

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Chapter 7: Networking with Neutron

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Neutron Architecture Overview

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Installing Neutron on the Controller

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Configuring the Network Node

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Provider Networks and External Connectivity

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Self-Service Tenant Networks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Virtual Routers, DHCP, and NAT

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Floating IPs and External Access

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Security Groups and Port Security

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

VXLAN, Geneve, and VLAN Backends

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Chapter 8: Block Storage with Cinder and Object Storage with Swift

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Cinder Architecture and Installation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Configuring Cinder Backends: LVM, NFS, Ceph RBD

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Managing Volumes, Snapshots, and Volume Types

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Swift Architecture Overview

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Installing and Configuring Swift

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Using Object Storage for Images, Backups, and Data

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Chapter 9: Storage Deep Dive: Ceph Integration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Why Ceph with OpenStack

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Ceph Architecture Primer

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Deploying a Ceph Cluster

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Integrating Ceph with Cinder

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Integrating Ceph with Glance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Integrating Ceph with Nova Ephemeral Storage

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Monitoring and Maintaining Ceph

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Chapter 10: The Dashboard, Orchestration, Load Balancing, and Containers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Installing and Configuring Horizon

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Using the Dashboard Effectively

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Heat Orchestration Service

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Octavia Load Balancer as a Service

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Magnum Container Orchestration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Manila Shared File Systems

Chapter 11: Daily Operations and Resource Management

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Managing Projects, Users, and Roles

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Creating Networks, Subnets, and Routers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Launching Instances with the CLI

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Managing Volumes, Images, and Key Pairs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Resource Quotas and Limits

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Instance Lifecycle: Snapshot, Resize, Migrate, Evacuate

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Scheduling Policies and Placement Filters

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Chapter 12: Security Hardening and Access Control

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Identity Security: Domains, Groups, and Fine-Grained Roles

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Policy Files and Authorization Rules

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

TLS Everywhere: Certificates and Encryption

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Securing the API Endpoints

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Network Isolation and Microsegmentation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Secrets Management and Credential Rotation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Firewalling and Network Security Best Practices

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Compliance and Auditing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Chapter 13: Automation, APIs, and Infrastructure as Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

OpenStack REST APIs Overview

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Using the OpenStack CLI Effectively

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Client SDKs: Python, Go, and Others

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Terraform Provider for OpenStack

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Ansible Modules and Automation Playbooks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

CI/CD Integration for Cloud Workloads

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Chapter 14: Monitoring, Performance, Upgrades, and Troubleshooting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Monitoring Architecture: Telemetry, Metrics, Logging

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Setting Up Alerting and Dashboards

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Capacity Planning and Performance Tuning

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Backup Strategies and Disaster Recovery

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Upgrading OpenStack Releases

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Systematic Troubleshooting: A Methodical Approach

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Common Failure Modes and Remediation Procedures

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Conclusion: Operating a Production OpenStack Cloud

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

Lessons from Production Deployments

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

The Path Forward for OpenStack Operators

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.

References

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/theopenstackhandbook.