

龙蜥操作系统（Anolis OS）8.9 介绍

龙蜥操作系统（Anolis OS）8.9 是由龙蜥社区（OpenAnolis）推出的开源 Linux 发行版的重要版本。以下是该操作系统的核心介绍：

1. 基本定位

龙蜥操作系统是一款支持多计算架构的企业级操作系统，100% 兼容 CentOS 8 的软件生态，同时兼容其他主流 Linux 发行版的操作习惯。它旨在为用户提供稳定、高性能、安全、可靠的底层运行环境，特别针对云端场景进行了深度优化。

2. 多架构支持

Anolis OS 8.9 提供了丰富的安装和部署形态，全面支持以下架构：

- x86_64：提供基础、精简、网络安装 ISO，以及 QEMU 虚拟机镜像和容器镜像。
- aarch64 (ARM)：提供对应的完整安装与精简镜像。
- loongarch64 (龙芯)：提供专属的基础、精简及网络安装 ISO 和虚拟机镜像。

3. 核心新特性与亮点

作为 Anolis OS 8 系列的第五个小版本，8.9 在软硬件协同方面进行了显著增强：

- 新硬件平台支持：全面支持海光四代 CPU 平台；同时支持 Intel 最新的 EMR 和 SPR 架构处理器，并增强了 QAT/DLB/IAA 等硬件加速能力 [[16]]。
- 安全与虚拟化增强：新增海光 CSV/CSV2 机密虚拟机启动/迁移及机密容器支持；新增 CSV3 安全虚拟机并支持热迁移功能。此外，社区虚拟机镜像现已支持 Legacy + UEFI 双启动模式。
- 内核与性能调优：ANCK 镜像的默认内核升级至 5.10.134-16.2 版本。系统内置了性能调优组件 KeenTune（默认安装但未启用，可按需替代 tuned），并提供 plugsched 组件以支持内核调度器子系统的热升级。
- 软件生态与合规性：增强了 glibc，实现了对新国标 GB18030-2022 的支持；新集成了阿里巴巴普惠体字体。同时，提供了 Dragonwell 8/11/17 三个版本的 Java 组件，以及基于 Clang/LLVM 开发的自研 C/C++ 编译器

(alibaba-cloud-compiler), 以更好地服务于大规模云业务场景 [[19]]。

- 工具链更新: ethtool 升级至 6.6 版本, 增强了对 CMIS 模块接收接口的支持, 并保持了与旧版本内核头文件的兼容性 [[19]]。

4. 适用场景

凭借其高度的 CentOS 兼容性和对国产芯片 (如海光、龙芯) 的深度适配, Anolis OS 8.9 非常适合用于企业级服务器、私有云/公有云基础设施、容器化部署环境, 以及对数据安全和国密合规有较高要求的业务场景。

系统 Kubernetes 集群接入与部署技术规范书

1. 项目背景与总体原则

本项目一期已建成并投产了基于 Kubernetes（以下简称 K8s）的容器化平台，容器运行时采用 `containerd`。后续项目的**所有**新增业务服务需无缝接入一期现有 K8s 集群。

为保障一期核心生产业务的绝对稳定，后续项目实施须遵循“零干扰、强隔离、严管控”的总体原则。

1.1 实施“红线”要求

1. 严禁升级/降级基础组件：不得以兼容性问题为由，要求升级或降级现有 K8s 集群版本、`containerd` 版本及 CNI/CSI 等核心插件版本。

2. 严禁引入 Docker 运行时：现有集群统一使用 `containerd` 作为容器运行时（CRI）。严禁在任何集群节点上安装 Docker 引擎，严禁修改 `containerd` 的全局配置文件。所有镜像必须符合 `containerd` 的运行规范。

3. 严禁修改集群全局配置：不得修改 `kube-proxy`、`CoreDNS`、`kube-apiserver` 等集群级核心组件的全局配置参数。

4. 纯 Pod 化部署：后续项目所有新增服务必须 100% 容器化，以标准 K8s 工作负载（Deployment/StatefulSet/DaemonSet 等）及 Pod 形式运行，严禁在物理机/虚拟机上以传统进程方式旁路部署业务服务。

2. 网络与端口防冲突专项规范

为避免后续项目服务与一期服务发生网络冲突，特别是 Service (SVC) 端口冲突，须严格执行以下网络隔离与端口规划方案：

2.1 命名空间强隔离

1. 独立 Namespace：后续项目业务服务必须部署在校方指定的独立 Namespace（如 `phase2-biz`）中，严禁将后续项目混部在一期业务的 Namespace 内。

2. 跨 Namespace 访问规范：后续项目需调用一期服务，首先要通过 K8s 内部 DNS 标准格式（`<svc-name>.<namespace>.svc.cluster.local`）进行访问，严禁通过硬编码 NodeIP 或 PodIP 进行访问。

2.2 Service 与端口防冲突机制

1. ClusterIP 管理: Service 的 ClusterIP 由 K8s 自动分配, 后续项目需确保业务代码中不依赖固定的 ClusterIP, 以防范集群重建或漂移导致的冲突。

2. NodePort 严格管控:

若后续项目服务必须暴露 NodePort, 须在实施前向校方提交《NodePort 端口使用申请清单》。校方将分配专用的 NodePort 端口段 (如 `32000-32100`) 给后续项目使用。严禁私自占用集群默认的端口段, 避免与一期服务冲突。

3. 禁用 HostPort: 原则上严禁使用 `hostPort`。如特殊业务 (如某些监控 Agent) 确需使用, 须提前报备, 并通过 Pod 反亲和性 (Anti-Affinity) 确保同一节点上不会调度多个占用相同 HostPort 的 Pod。

4. 统一出口: 后续项目对外暴露的 HTTP/HTTPS 服务, 须统一接入集群现有的 Ingress Controller 或 API 网关。通过配置不同的 Host 域名或 Path 路径进行路由分发, 严禁为后续项目服务单独绑定宿主机端口。

2.3 DNS 与域名冲突避免

1. 需梳理后续项目服务的 Service 名称, 确保在后续项目 Namespace 内不重名。

2. 若需配置 Ingress 域名, 须使用校方规划的后续项目专属子域名, 严禁占用一期业务的域名及泛解析规则。

3. 资源调度与存储防冲突规范

3.1 资源配额与限制

1. ResourceQuota 与 LimitRange: 校方将在后续项目 Namespace 配置资源配额。后续项目编写的所有 Pod YAML 须配置 `resources.requests` 和 `resources.limits` (包括 CPU 和 Memory)。未配置 Limits 的 Pod 将被准入控制器 (Admission Controller) 拦截。

2. QoS 保障: 核心服务需配置为 `Guaranteed` 级别 (requests 等于 limits), 防止在节点资源紧张时被 K8s 驱逐 (Evict), 从而影响一期服务。

3.2 存储卷规范

1. 命名规范: 后续项目申请的所有 PVC 名称须带有后续项目前缀 (如

`pvc-phase2-db-xxx`), 避免与一期 PVC 名称冲突。

2. StorageClass 使用: 须使用集群现有的 StorageClass, 严禁私自创建新的 StorageClass 或修改现有 StorageClass 的底层存储参数。

3. 数据目录隔离: 若使用 HostPath (极不推荐, 需特批), 须严格限定在校方规划的后续项目专属宿主机目录下, 严禁读写一期业务的数据目录。

4. 安全与权限管理规范

1. RBAC 最小权限原则: 需为后续项目服务配置独立的 ServiceAccount, 并绑定最小化权限的 Role/RoleBinding。严禁使用 `cluster-admin` 或绑定 `default` ServiceAccount。

2. 镜像安全: 后续项目业务镜像须推送至甲方指定的私有镜像仓库 (Harbor 等), 严禁在生产环境直接通过公网 `docker pull` 拉取镜像, 以防供应链攻击及网络超时。

3. 安全上下文: Pod 必须配置 `runAsNonRoot: true`, 严禁以 root 用户运行容器进程; 需开启 `readOnlyRootFilesystem: true` (如业务允许)。