Akash Network

Running Containers in a Distributed System

The views, thoughts, and opinions expressed in this presentation are solely those of the individual author and do not necessarily reflect the official policy or position of PennyMac

Nathan Moore
VP, App Dev

PENNYMAC

Usual Disclaimer

The views, thoughts, and opinions expressed in this presentation are solely those of the individual author and do not necessarily reflect the official policy or position of PennyMac

This presentation was made with assistance from Google's Nano Banana

Nathan Moore
VP, App Dev
Pennymac

akash

The name “Akash” comes from Sanskrit:

ākāśa (आकाश)

“the basis and essence of all things in the material world; the first element created.”

–Wiki

Meaning of Akash

https://en.wikipedia.org/wiki/Akasha

akash

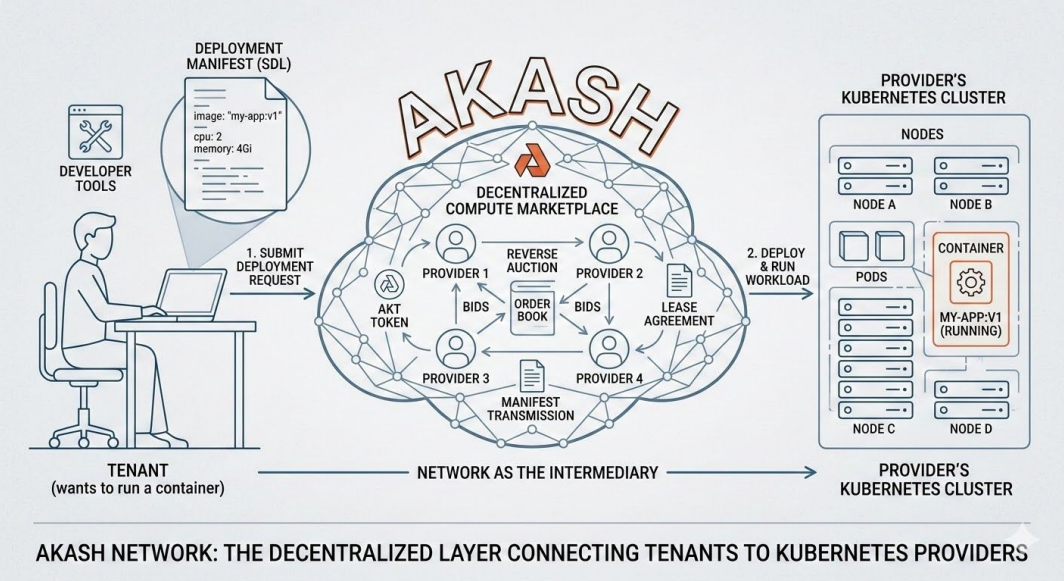
Akash is an open source project, licensed under Apache License 2.0

There is also a governance structure provided by a decentralized community, with decisions made by AKT token holders

The original creator was Overclock Labs, which remains the largest contributor to the project

---

Akash Network is the middle layer between people who want to run a container and people who want to host a container



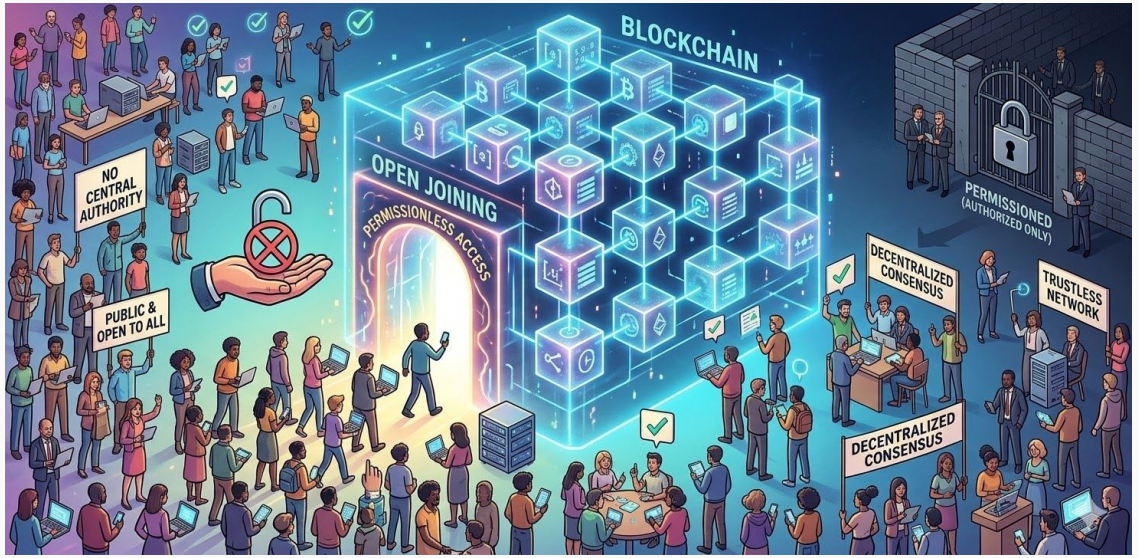
Akash Network is the middle layer between people who want to run a container and people who want to host a container

Anyone can participate to provide compute resources

Anyone can participate to consume compute resources

This requires “permissionless” access on both sides

It also requires “trustlessness” with strong encryption for secure compute, where compute jobs cannot be corrupted



The mission is to democratize both usage and provision of compute resources

- Application Containers, $3.5Bn Revenue in 2025
- 27% CAGR growth, expect $39.1Bn Revenue in 2035
- Cloud vs. On-Prem hosting
  - 75.5% vs. 24.5%

Global Application Container Marketing Notes

https://www.grandviewresearch.com/industry-analysis/cloud-computing-industry
https://www.researchnester.com/reports/application-container-market/8266#:~:text=Application%20Container%20Market%20Outlook:.evaluated%20at%20USD%204.5%20billion

- Founded in 2015
  - Greg Osuri and Adam Bozanich
- Mainnet launch in 2020
- Utilizes Cosmos SDK
- GPU support in 2023
- Active Discord:
  - https://discord.com/invite/akash
  - About 24k people in Discord

Akash Network History

akash

Network Summary

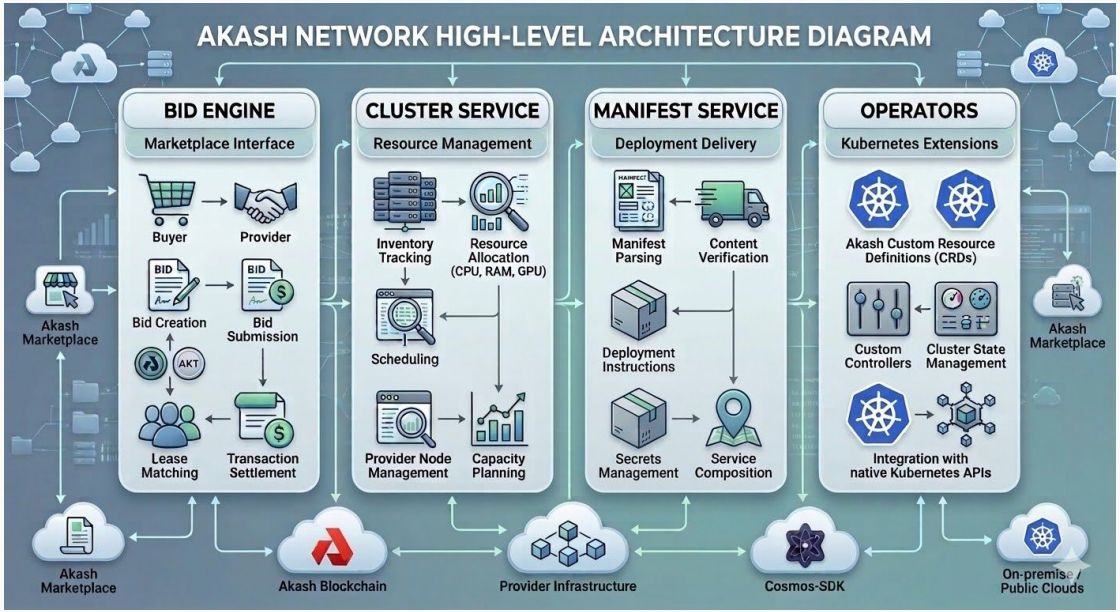
USD spent (24h)	Total spent USD	New leases (24h)	Total leases
$2.17K ↓ 11.27%	$4.95M ↑ 0.04%	240 ↓ 28.36%	428K ↑ 0.06%
Graph	Graph	Graph	Graph

Resources leased

Active leases	Compute	Graphics	Memory	Storage
516 ↓ 7.86%	1.94K CPU ↓ 25.65%	70 GPU ↑ 22.81%	4.83 TB ↑ 5.35%	20.33 TB ↑ 1.07%
Graph	Graph	Graph	Graph	Graph

Network Capacity

Active providers	Compute	Graphics	Memory	Storage
53 ↓ 1.85%	8.2K CPU ↓ 4.65%	182 GPU ↓ 1.09%	52.95 TB ↓ 1.6%	514.65 TB ↓ 5.74%
Graph	Graph	Graph	Graph	Graph


AKASH NETWORK HIGH-LEVEL ARCHITECTURE DIAGRAM
BID ENGINE
Marketplace Interface
Buyer
Provider
Bid Creation
Bid Submission
AKT
Lease Matching
Transaction Settlement
CLUSTER SERVICE
Resource Management
Inventory Tracking
Resource Allocation (CPU, RAM, GPU)
Scheduling
Provider Node Management
Capacity Planning
MANIFEST SERVICE
Deployment Delivery
Manifest Parsing
Content Verification
Deployment Instructions
Secrets Management
Service Composition
OPERATORS
Kubernetes Extensions
Akash Custom Resource Definitions (CRDs)
Custom Controllers
Cluster State Management
Integration with native Kubernetes APIs
Akash Marketplace
Akash Marketplace
Akash Marketplace
Akash Blockchain
Provider Infrastructure
Cosmos-SDK
On-premise / Public Clouds

Bid Engine

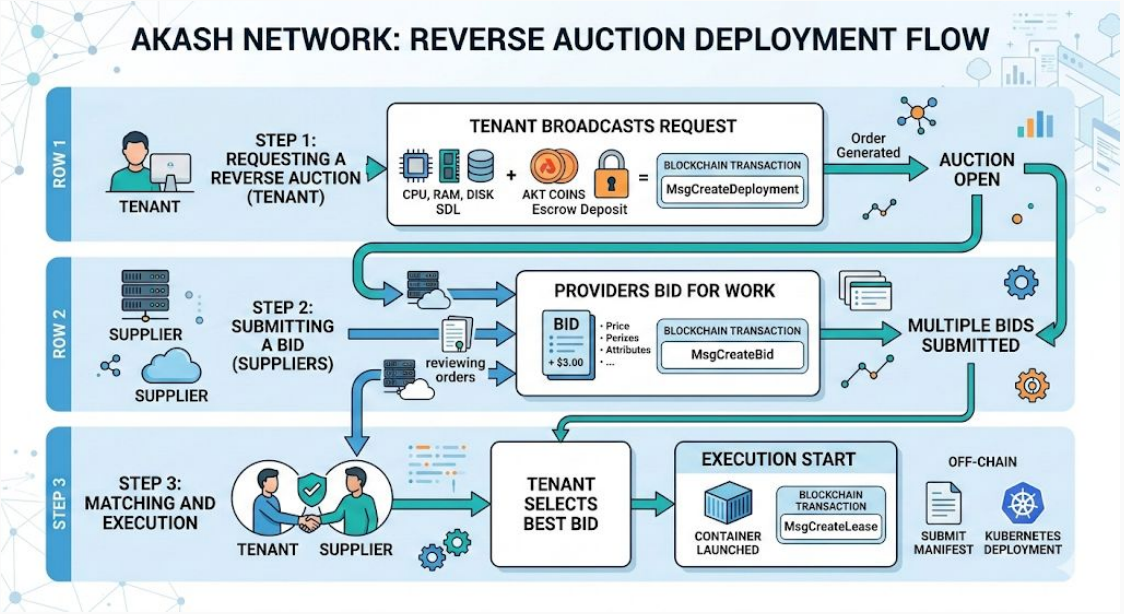
akash

Reverse Auction

An auction where multiple providers all submit bids for how much they are want to complete the service, while a single buyer chooses which bid to accept

How It Works: Reverse Auction

akash



When someone (tenant) wants to have their container run, on request, all available providers who want to bid on running this container submit a bid, and the buyer chooses which one, after which container execution starts

akash

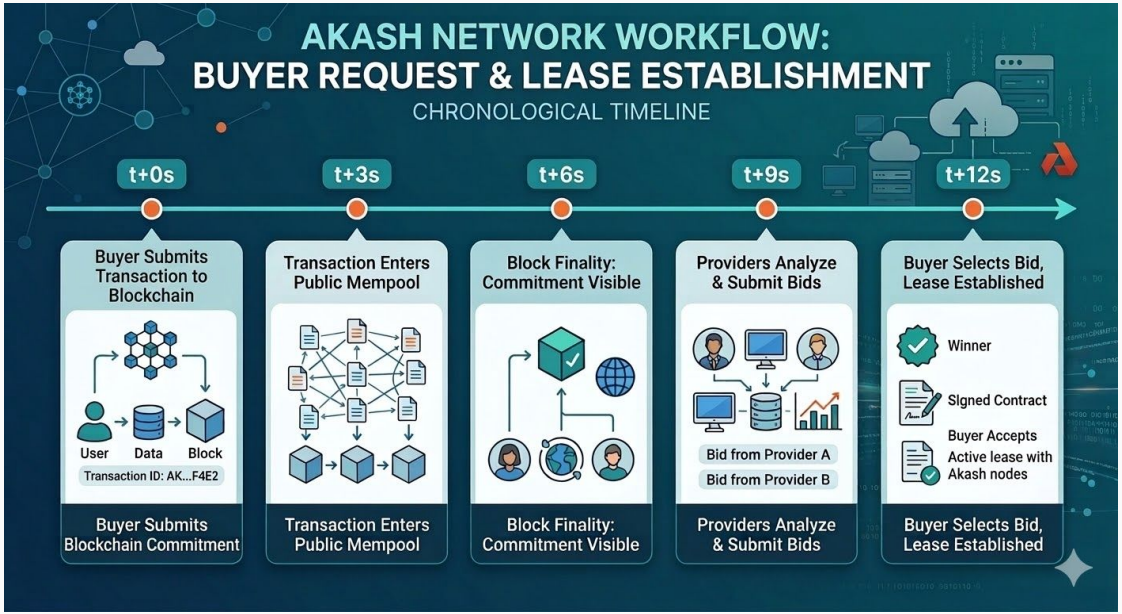
Blockchain semantics
Every participant has a unique wallet address, and can sign transactions to cryptographically validate them.

1. MsgCreateDeployment (Tenant commits transaction)
2. MsgCreateBid (Suppliers commit transactions)
3. MsgCreateLease (Tenant signs transaction of chosen supplier)

Buyer (Tenant) and Supplier (Provider)

- Cosmos SDK
    - 6 second blocktime
    - Gossip protocol
    - Consensus protocol
        - Tendermint
            - Byzantine Fault Tolerance
            - Instant finality
    - Providers can run a Validator node
        - Proof-of-Stake

Blockchain Coordination



Timeline from Lease Request to Lease Establishment

- Public Ledger
  - Every bid, pricepoint, and hardware promise is permanently recorded and readable by all
- No gatekeepers
  - Anyone can deploy or provide resources globally

This is part of “Permissionlessness”, where providers cannot engage in price discrimination or prevent others from competing, as all competitors have full visibility into the market clearing price

---

Transparency and Fairness

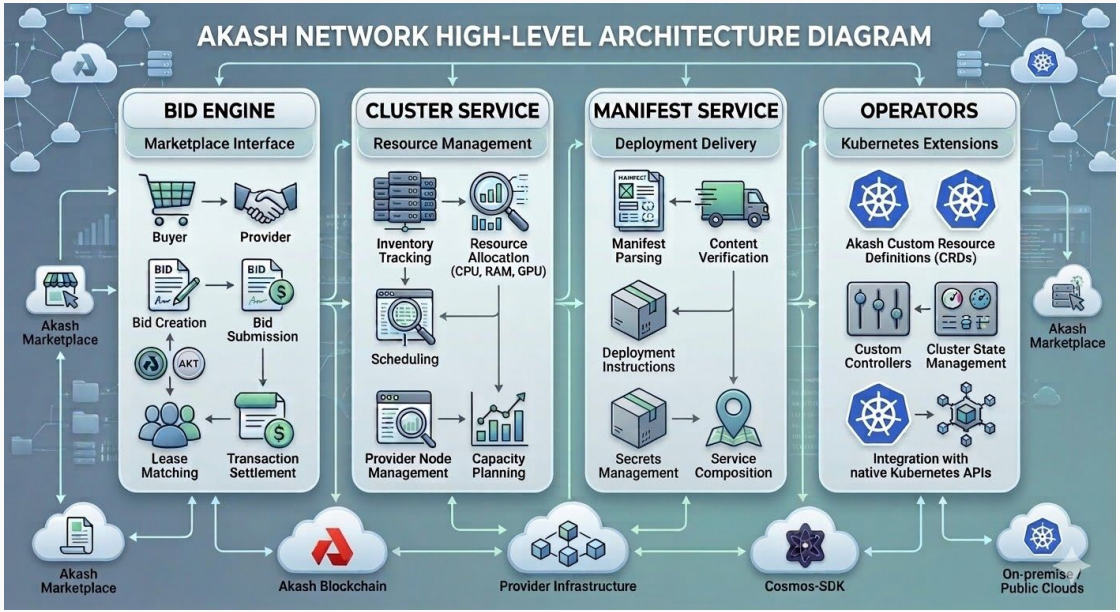
- To pay for compute, rent is due every block cycle (6 seconds)
  - This is usually a fraction of a penny (~$0.0004)

Settlement occurs on every block instead of a monthly billing cycle. Termination of lease requires 1 block to elapse, ceasing billing immediately after

This delivers a flexible billing, pay-as-you-use model, which either side can cancel at any time

---

Rent is due



Cluster Service

akash

Provider-Side cluster management

- Choose your hardware setup
  - Minimum requirements
  - Availability
  - Resource Allocation
  - Where/how you want to host
  - Network

Cluster Service

akash

Minimum requirements:

Single Server (Worker & Control Plane Combined)

- x86_64 only
- CPU 8 cores
- RAM 16GB
- Storage 150GB SSD
- Internet connectivity
- Dynamic DNS or static public IP

Cluster Service

https://akash.network/docs/providers/getting-started/hardware-requirements/

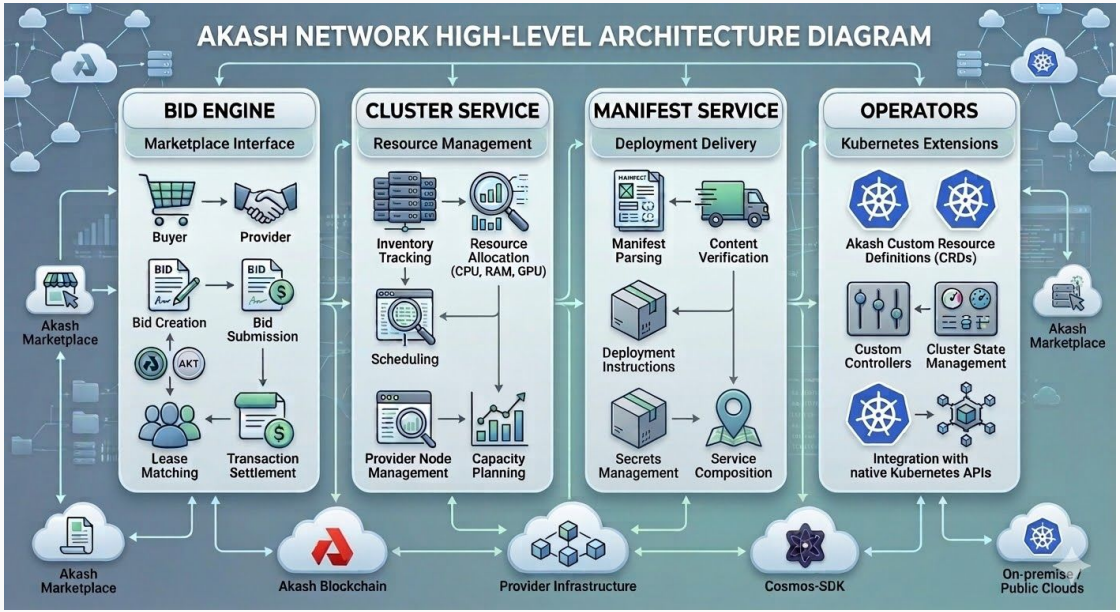
Deploy your cluster via Ansible Playbook, Kubespray, or Provider Console

Requirements

- Ubuntu 24.04 LTS
- Root or sudo access on all nodes
- SSH access to all nodes

Cluster Service

https://akash.network/docs/providers/setup-and-installation/provider-playbook/



Manifest Service

akash

Stack Definition Language (SDL)

Akash's custom YAML-based declarative language used to define:

- Configuration
- Resource requirements
- Deployment parameters

Configures a "Manifest" file, similar to a Docker Compose file

---

Manifest Service

https://akash.network/docs/providers/setup-and-installation/provider-playbook/

akash

SDL example, OpenClaw

README | View SDL

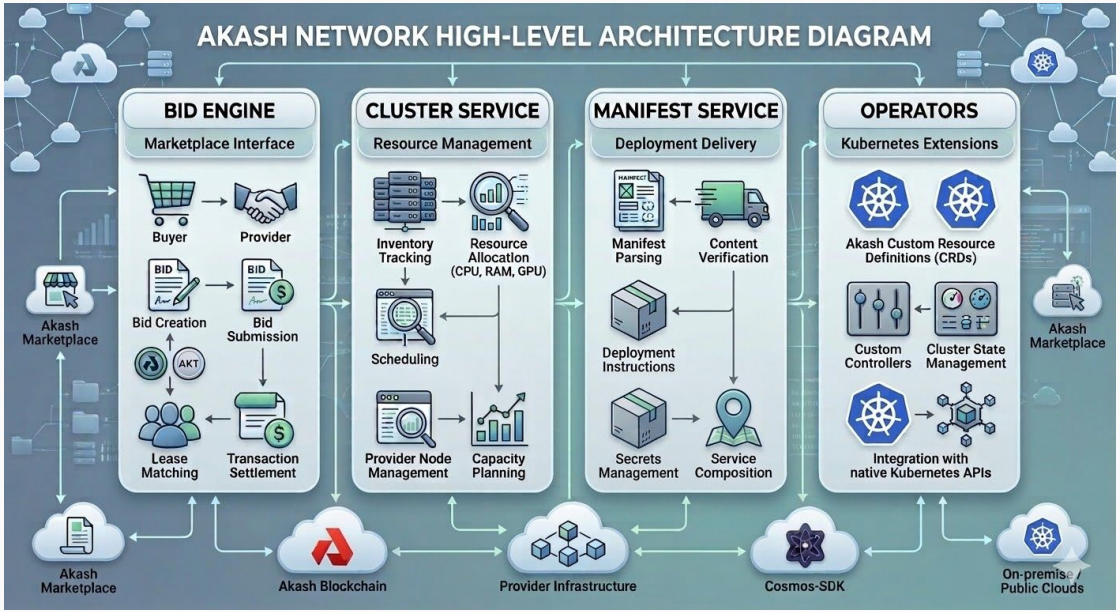
OpenClaw

Deploy

```
---
version: "2.0"
services:
  openclaw:
    image: ghcr.io/zjuuu/openclaw-docker:openclaw-v2026.2.12@sha256:54078e1fc41c781a94f48846cf286f18d0632a4e49c69d78aee799ba938fffa0
    env:
      # REQUIRED: Set a strong password for the setup UI
      - SETUP_PASSWORD=CHANGE_ME_TO_STRONG_PASSWORD
      - OPENCLAW_CONFIG_PATH=/data/.openclaw/openclaw.json
    expose:
      - port: 8080
        as: 80
        to:
          - global: true
        http_options:
          max_body_size: 104857600
```

Manifest Service

https://console.akash.network/templates/akash-network-awesome-akash-openclaw


AKASH NETWORK HIGH-LEVEL ARCHITECTURE DIAGRAM
BID ENGINE
Marketplace Interface
Buyer
Provider
Bid Creation
Bid Submission
AKT
Lease Matching
Transaction Settlement
CLUSTER SERVICE
Resource Management
Inventory Tracking
Resource Allocation (CPU, RAM, GPU)
Scheduling
Provider Node Management
Capacity Planning
MANIFEST SERVICE
Deployment Delivery
Manifest Parsing
Content Verification
Deployment Instructions
Secrets Management
Service Composition
OPERATORS
Kubernetes Extensions
Akash Custom Resource Definitions (CRDs)
Custom Controllers
Cluster State Management
Integration with native Kubernetes APIs
Akash Marketplace
Akash Marketplace
Akash Marketplace
Akash Blockchain
Provider Infrastructure
Cosmos-SDK
On-premise / Public Clouds

Operators

akash

Inventory Operator

- Discovers and tracks cluster resources (CPU, GPU, memory, storage) in real time
  - Supports hardware feature discovery (GPU models, storage classes)
  - Integrates with Nvidia device plugin for GPUs
  - Updates K8s node labels with capabilities
  - Publishes inventory to provider service for bidding

Operators

https://akash.network/docs/providers/architecture/operators/

IP Operator

- Manages static IP address allocation for deployments via MetalLB integration
  - Allocates dedicated IPs for deployments
  - Supports IP sharing across services (same tenant)
  - Tracks IP pool usage and availability
  - Handles IP lifecycle (create, update, delete)
  - Exposes HTTP API for monitoring

Operators

https://akash.network/docs/providers/architecture/operators/

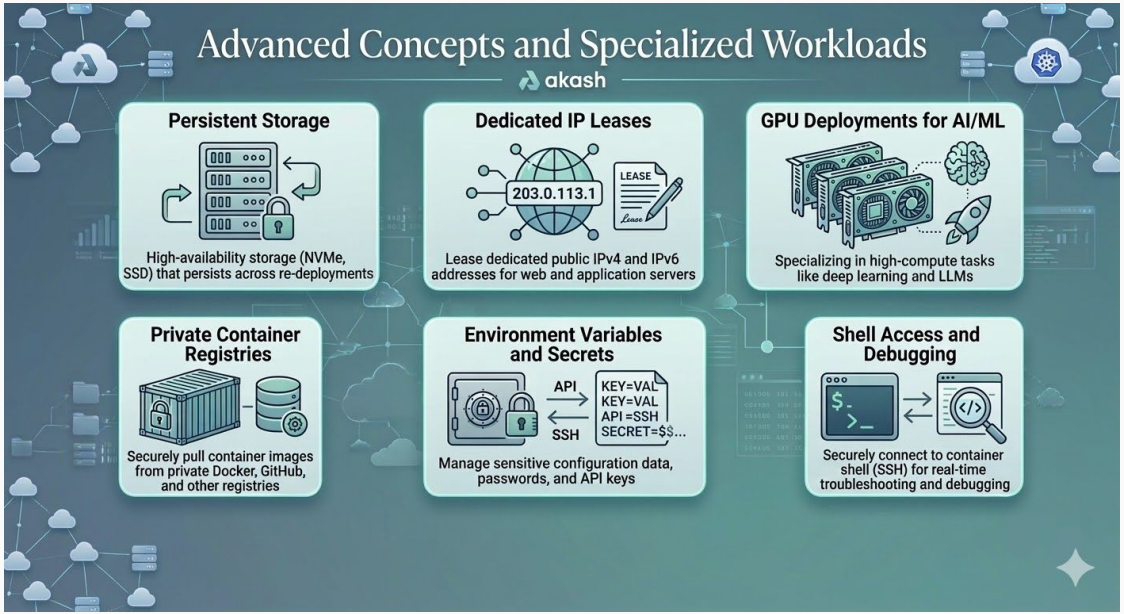
akash

Hostname Operator

- Manages custom hostname assignments and creates ingress resources for deployments
  - Integrates with Ingress controllers
  - Handles hostname conflicts and validation
  - Supports wildcard hostnames
  - Monitors hostname health

Operators

https://akash.network/docs/providers/architecture/operators/



Advanced Concepts and Specialized Workloads

https://akash.network/docs/learn/core-concepts/

akash

Default storage is ephemeral

- Lost on container restart

If provider supports it, can deploy to persistent storage instead.
Note, persistent storage is provider-local. Switching providers loses data

There are three storage classes:

1. NVMe storage (“Beta3”)
2. SSD storage (“Beta2”)
3. HDD storage (“Beta1”)

Persistent Storage

https://akash.network/docs/learn/core-concepts/persistent-storage/

Default deploys use shared IPs, mapped via provided Hostname

Custom IP leases allow

- Custom domains
- Static IPs
- non-HTTP services

IP leases are available on a per-Provider basis, at additional cost.

IP Leases

https://akash.network/docs/learn/core-concepts/ip-leases/

akash

GPUs are available on a per-provider basis

Tenants can request specific GPU models including multiple GPUs, and match against what providers have available on lease instantiation

Providers offer H200 on down, at different price points

Supports CUDA runtime

GPU Deployments for AI/ML

https://akash.network/docs/learn/core-concepts/gpu-deployments/

Specify private container registry credentials inside of SDL

Any Docker-compatible private registry is supported, including

- ECR
- GCR
- GHCR
- etc

Private Container Registries

https://akash.network/docs/learn/core-concepts/private-containers/

Environmental variables are set within SDL

Use Secret Management Tools

- AWS Secrets Manager
- HashiCorp Vault
- Google Secret Manager
- Azure Key Vault

Have application fetch at runtime, use SDL to specify which tool

---

Environmental Variables and Secrets

https://akash.network/docs/learn/core-concepts/environment-secrets/

Prerequisites:

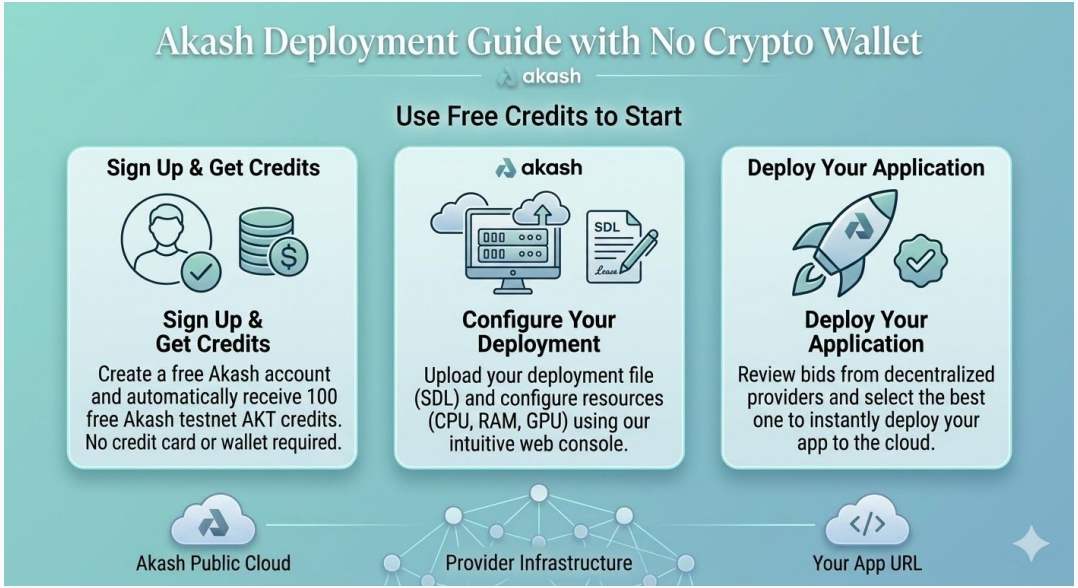
- Active deployment with a lease
- Akash CLI (provider-services) installed
- Wallet public key accessible
- Deployment Sequence Number (DSEQ)
- Provider address

Use the “provider-services” package (the Akash CLI) to authenticate and access the remote container

---

Shell Access and Debugging

https://akash.network/docs/learn/core-concepts/shell-access/

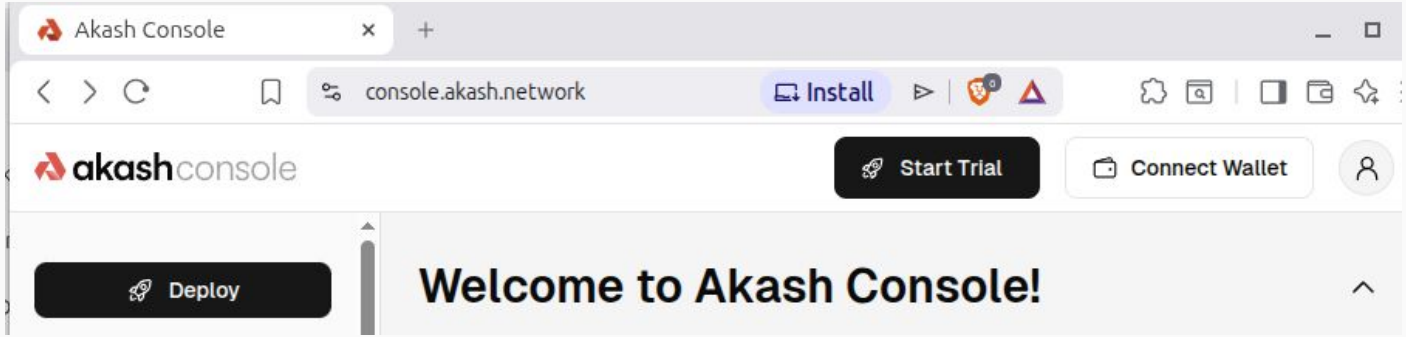


Deployment Guide – No Crypto Wallet

https://akash.network/blog/introducing-credit-card-payments-in-akash-console/

https://console.akash.network

Select “Start Trial”

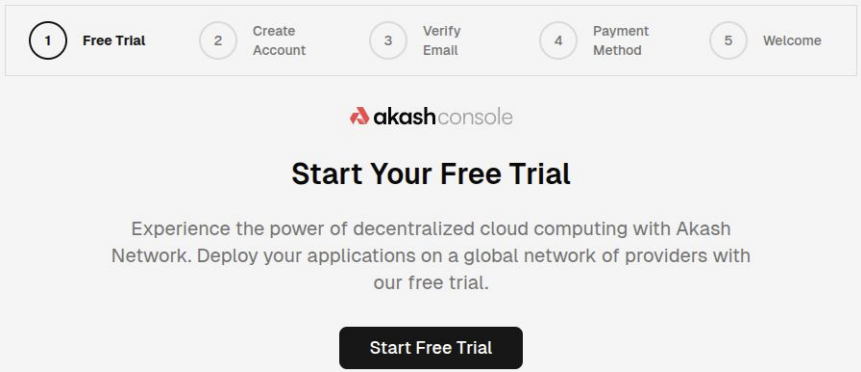


Deployment Guide – No Crypto Wallet

https://akash.network/docs/learn/core-concepts/shell-access/

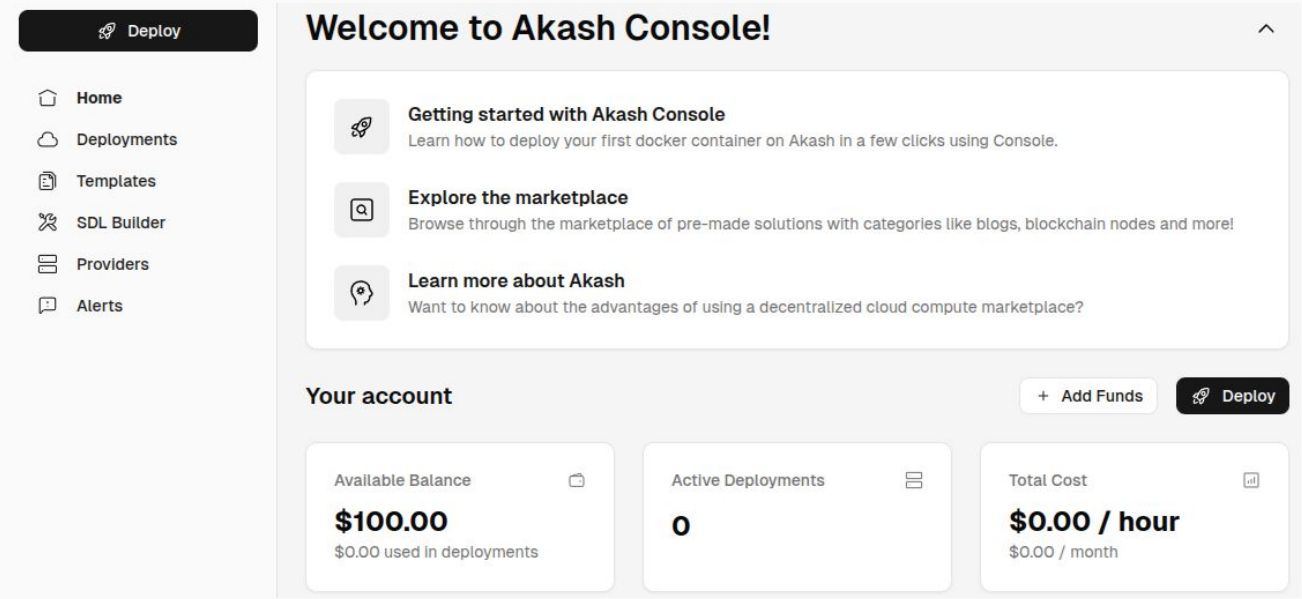
https://console.akash.network

Follow instructions



Deployment Guide – No Crypto Wallet

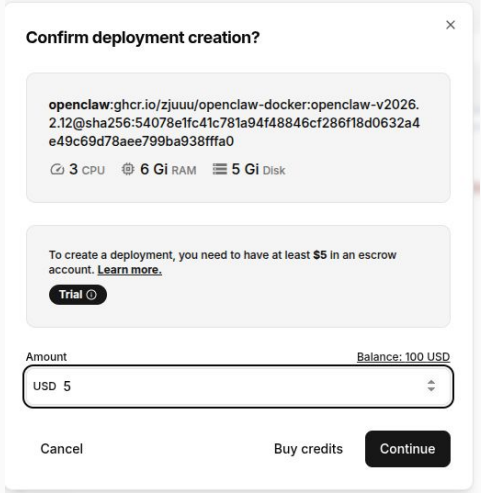
https://akash.network/docs/learn/core-concepts/shell-access/

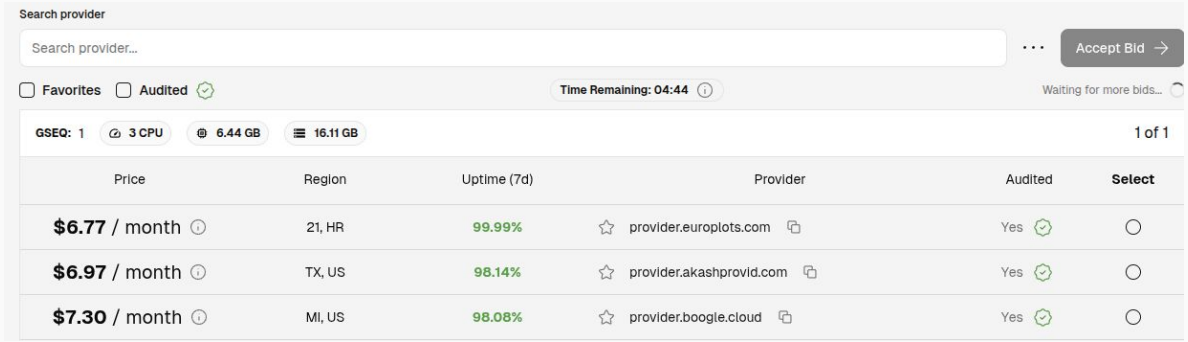


Deployment Guide – No Crypto Wallet

https://akash.network/docs/learn/core-concepts/shell-access/

Deploy OpenClaw



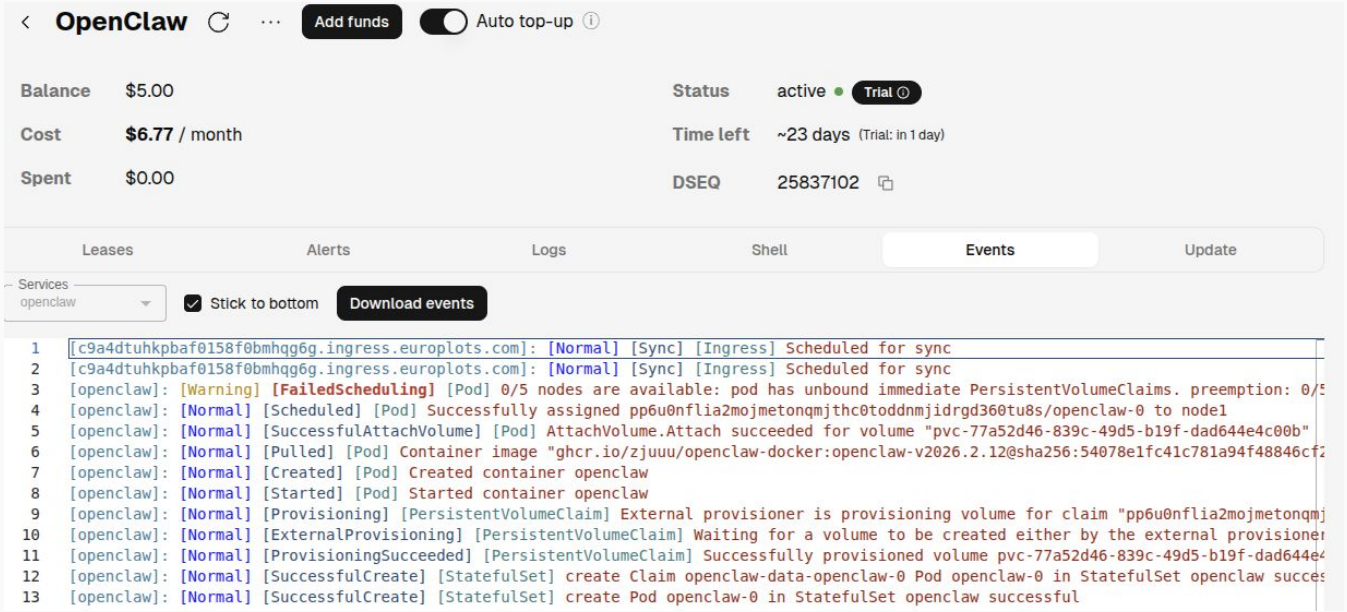


Deployment Guide – No Crypto Wallet

https://akash.network/docs/learn/core-concepts/shell-access/

akash

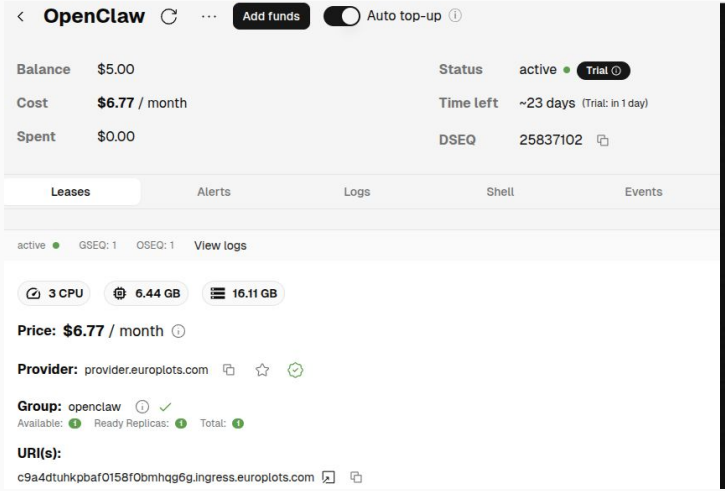
Success

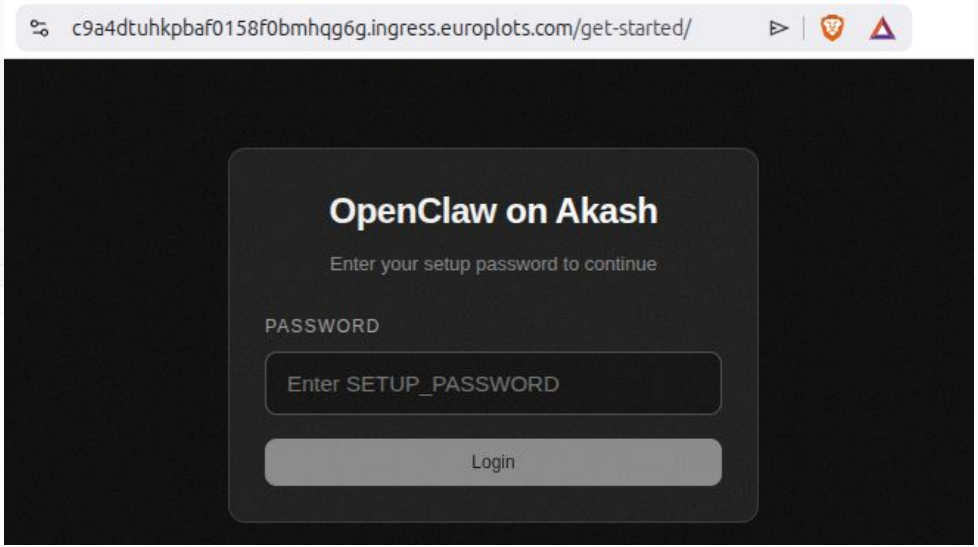


Deployment Guide – No Crypto Wallet

https://akash.network/docs/learn/core-concepts/shell-access/

OpenClaw Details





Deployment Guide – No Crypto Wallet

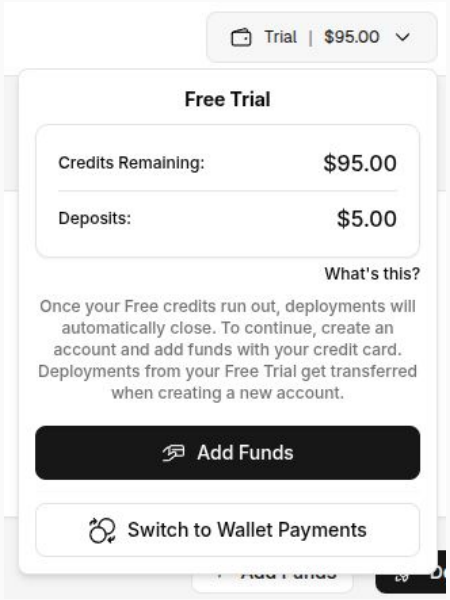
https://akash.network/docs/learn/core-concepts/shell-access/

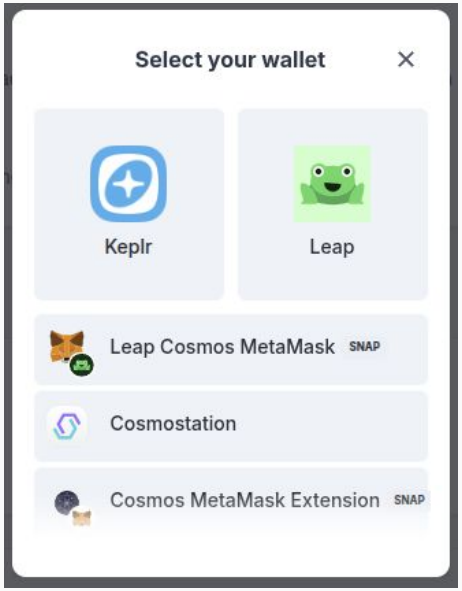
Using Akash with Crypto Wallet

akash

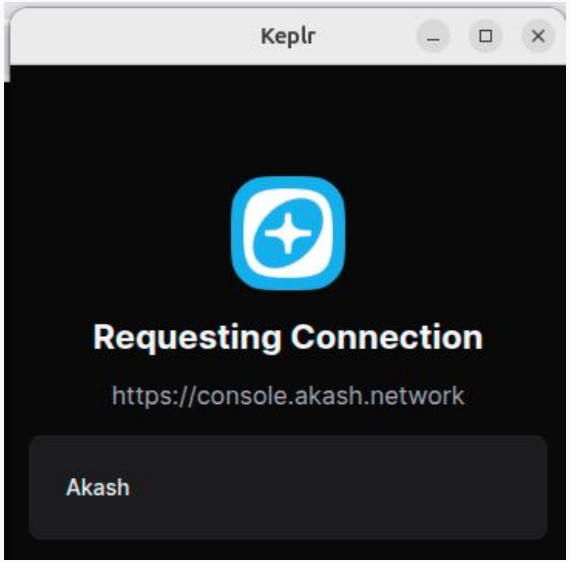
Switch to Wallet Payments

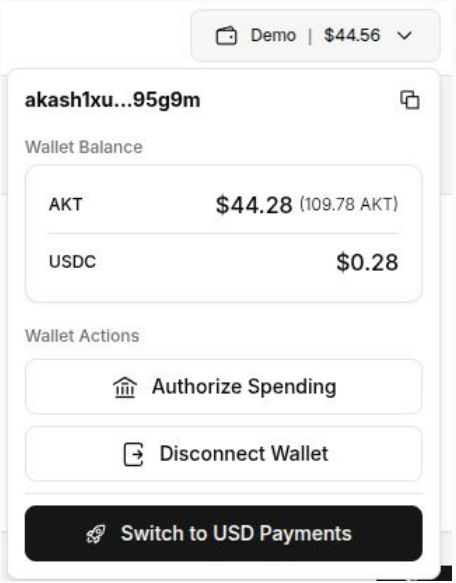
- Work natively with AKT
- Gain greater control
- Exit free trial so deploys last longer than 24 hours





Using Akash with Crypto Wallet





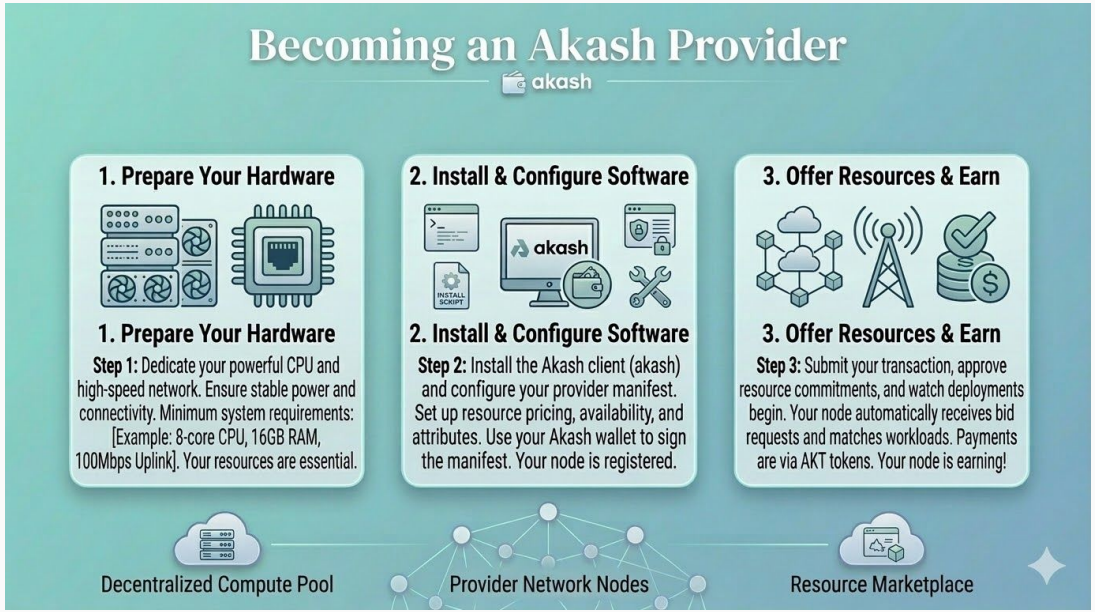
Using Akash with Crypto Wallet

akash

DeFi semantics

- Install preferred wallet (Keplr)
- Fund your wallet with ATOM (Cosmos coin)
  - Fund from On-ramp (CeFi or other)
- Bridge ATOM to Osmosis
- Use Osmosis DEX to swap ATOM for AKT
- Withdraw AKT back from Osmosis to Keplr

Using Akash with Crypto Wallet



Becoming an Akash Provider

https://akash.network/docs/providers/setup-and-installation/provider-playbook/

Hardware:
Intel I7700K (8 core)
16GB RAM
1TB NVMe

OS:
Ubuntu 24.04 LTS

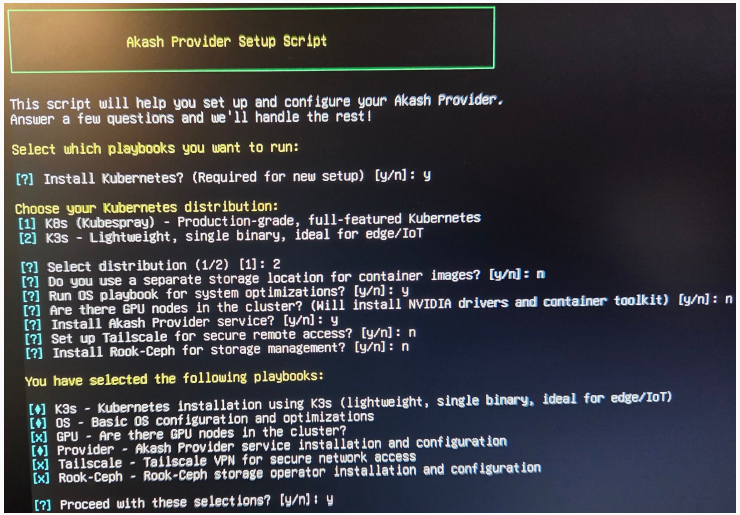
Becoming an Akash Provider

https://akash.network/docs/providers/setup-and-installation/provider-playbook/

akash

Install via provided Ansible Playbook

- Interactive guide
- Flexible config
- Infrastructure as Code (configs versioned)



Becoming an Akash Provider

https://akash.network/docs/providers/setup-and-installation/provider-playbook/

akash

Utilize an automated pricing script to optimize pricing

Akash provider pricing script with per model GPU pricing

price_script_generic.sh

Raw

```
#!/bin/bash
# WARNING: the runtime of this script should NOT exceed 5 seconds! (Perhaps can be amended via AKASH_BID_PRICE_SCRIPT_PROCESS_TI
# Requirements:
# curl jq bc mawk ca-certificates
# Version: Sept-08-2023
set -o pipefail

# # Specify the output file path in the /tmp directory
# var_output_file="/tmp/env_variables.txt"

# # Loop through all environment variables and append them to the output file
# for var in $(printenv); do
#   echo "$var" >> "$var_output_file"
# done
```

Becoming an Akash Provider

https://gist.github.com/chainzero/33978bf221eb35f10a7392ed9bae8caa

Questions?