Decentralized Cloud Architecture Documentation

Overview

This document provides a detailed technical explanation of the Decentralized Cloud Architecture diagram used to provision a decentralized compute and storage environment using Infrastructure as Code (IaC).

It includes:

- Selected tech stack
- Terraform-based infrastructure provisioning strategy
- Network topology and flow
- Security practices
- Observability stack
- Future improvements

Selected Tech Stack

1. Decentralized Compute

- **Akash Network**: A decentralized marketplace for deploying and running containerized workloads.
    - *Why*: Akash is an open network that facilitates the secure and efficient buying and selling of computing resources.
- **Kubernetes (on Akash/Simulated)**: Used to orchestrate applications. The Akash node runs workloads using containerization similar to Kubernetes.
    - *Why*: Akash cloud provides helm chart to deploy cluster.

2. Decentralized Storage

- **IPFS (InterPlanetary File System)**: Peer-to-peer protocol to store and share data in a distributed file system.
- **Filecoin**: Persistent decentralized storage network that integrates with IPFS.
    - *Why*: Combines accessibility (IPFS) with permanence and incentivization (Filecoin).

- IPFS is a peer-to-peer protocol for distributing content across a network, while Filecoin is a blockchain-based, incentivized storage network built on top of IPFS.

3. API Gateway

- **NGINX / Envoy**: Used as a reverse proxy and gateway interface.
  - *Why*: High-performance routing, TLS termination

4. Monitoring & Observability

- **Prometheus**: Scrapes metrics from IPFS, Filecoin, and Akash nodes.
- **Grafana**: Visualizes Prometheus data with rich dashboards.
- **Node Exporter**: Used to expose host-level metrics.
  - *Why*: Proven stack for real-time monitoring and alerting.

5. IaC Tool

- **Terraform**
  - *Why*: Declarative, modular, cloud-agnostic infrastructure provisioning with strong ecosystem support.
  - *Reference*: GitHub repo - [Yashwanth-tss/decentralized-cloud]

How the IaC (Terraform) Scripts Provision Infrastructure

✅ **main.tf**

This is the entry point that includes:

```
module "akash" {
  source = "./modules/akash"
}

module "IPFS" {
  source = "./modules/IPFS"
}
```

```
module "filecoin" {
  source = "./modules/filecoin"
}

module "monitoring" {
  source = "./modules/monitoring "
}
```

Each module provisions a key part of the decentralized platform:

🖌️ **Modules**

- compute:
  - Deploys Akash-compatible nodes (on Akash or EC2 simulation).
  - Optionally provisions Kubernetes or Nomad if orchestrators are used locally.
  - Installs required Docker/Containerd runtimes and Akash deployment agents.
- storage:
  - Deploys IPFS nodes.
  - Connects to persistent Filecoin storage through Lotus clients or public gateways.
- monitoring:
  - Installs Prometheus, Grafana, and Node Exporter.
  - Sets up scraping configurations and dashboard provisioning.

Network Topology and Traffic Flow

1. Ingress

- Users interact via CLI/SDK or a Web UI.
- Requests hit the **API Gateway (NGINX/Envoy)**.

2. Routing

- The gateway routes requests to either:
  - **Akash Node (Kubernetes Pod)** for compute

    - **IPFS Node** for storage

3. Storage Access

- IPFS node interacts with Filecoin for data persistence.
- Compute workloads read/write to IPFS via native or gateway integration.

4. Monitoring Flow

- Prometheus scrapes metrics from:
    - IPFS, Filecoin, Akash nodes
    - Host OS metrics via Node Exporter
- Grafana fetches Prometheus data and visualizes dashboards.

Security Measures

Access Control

- IAM roles and policies control provisioning access.
- SSH keypairs and Akash wallet keys are stored securely (e.g., Secrets Manager).

Network Security

- Security groups and VPCs isolate services.
- API Gateway is the only public entrypoint.
- TLS termination is enforced at the gateway.

Secrets Management

- Environment variables, .tfvars, and encrypted files are used for secrets.
- Optionally integrate HashiCorp Vault or SOPS for advanced security.

Scaling Considerations

Compute

- Akash supports decentralized horizontal scaling by deploying workloads across provider networks.
- Simulated environments can use auto-scaling EC2/Nomad/K8s clusters.

Storage

- IPFS nodes can be horizontally scaled with DHT coordination.
- Filecoin miners can be added for redundancy.

Monitoring

- Prometheus scrapers and federation enable scale-out monitoring.
- Grafana supports multiple data sources and dashboard templating.

Observability Tools Used

Component	Tool	Purpose
Metrics	Prometheus	Scraping node/application metrics
Dashboards	Grafana	Visualizing system performance
OS Metrics	Node Exporter	Collecting CPU/RAM/disk stats

Note: Dashboards are provisioned via JSON definitions (datasources.json, dashboards.json).

Future Improvements

- **Service Mesh**: Add Istio/Linkerd for zero-trust communication.
- **KMS Integration**: Leverage Vault or AWS KMS for better secrets lifecycle.
- **Decentralized Logging**: Integrate Loki and Promtail or decentralized log persistence via IPFS.
- **CI/CD**: Automate SDL/manifest deployments using GitHub Actions.
- **Hybrid Orchestration**: Add Nomad as a lightweight alternative to Kubernetes.
- **Storage Enhancements**: Use Arweave or Crust Network for long-term storage.