Running Blockchain as a Service (BaaS) on Kubernetes

Shan Yu, Alibaba Cloud
AGENDA

1. Introduction to Blockchain and BaaS
2. Combining Blockchain with Docker and Kubernetes
3. Key Problems
4. Blockchain Application Scenario and Pattern
5. Demo
Introduction to Blockchain and BaaS

- **What is Blockchain**
  - Narrow sense: Distributed shared ledger technology, based on smart contract, reaching consensus on transactions among participants, with temper-proof transaction history in ledger
  - Broader sense: A new schema and collaboration model for constructing distributed trust network, connecting trusted data and achieving value flow among organizations, individuals and machines

- **What is BaaS**
  - "Blockchain as a Service" - Blockchain platform service on top of cloud, providing capabilities like deployment, operation, governance of blockchain system, and running and management of blockchain applications

[Diagrams of Centralized System (E.g. Traditional system, private chain), Decentralized System (E.g. Public chain), Multi-centralized System (E.g. Consortium chain)]
Combining Blockchain with Container and Kubernetes

- Characteristics of Blockchain
  - Blockchain system: Data-oriented, highly distributed, full-mesh network, long-running, complex system type
  - Blockchain applications: No standard, various types of applications

- Advantages of Blockchain in Docker
  - Standard packaging and distribution model
  - Uniform running environment and decoupling infrastructure

- Advantages of Blockchain on Kubernetes
  - Resource scheduling
  - Operation management
  - Extensive support for application types
  - Microservice architecture
  - Integration with cloud
  - Security and isolation
  - Community and ecosystem
History of Alibaba Cloud Blockchain

2017.10: Container Service Blockchain Solution (On Swarm)
2017.12: Container Service Blockchain Solution (On Kubernetes)
2018.7: BaaS – Hyperledger Fabric (On Kubernetes)
2018.9: BaaS – Ant Blockchain (On Kubernetes)
Alibaba Cloud Container Service for Kubernetes (ACK)

- **Computing**
  - ECS, EBM, GPU, FPGA

- **Network**
  - VPC, ENI, SLB, DNS

- **Storage**
  - EBS, NAS, CPFS, OSS

- **Managed Kubernetes**
  - Multi-cluster Management
  - Security Compliance
  - Hybrid Cloud Multi Cloud
  - Elasticity

- **Serverless Kubernetes**
  - Application Lifecycle Management
  - Integration

- **DevOps**
  - Gitlab
  - Jenkins
  - Aone

- **Microservices / Service Mesh**
  - Dubbo
  - SpringCloud
  - Istio

- **Enterprise Application Modernization**
  - .net
  - Java
  - Enterprise

- **Innovation**
  - AI
  - Blockchain
  - IoT

- **Application Modernization**
  - SpringCloud

- **DevOps Tools**
  - Gitlab
  - Jenkins
  - Aone

- **Microservices**
  - Dubbo
  - SpringCloud
  - Istio

- **Enterprise Applications**
  - .net
  - Java
  - Enterprise

- **Innovations**
  - AI
  - Blockchain
  - IoT

- **Public Cloud**
- **Dedicated Cloud**
Alibaba Cloud BaaS - Hyperledger Fabric Deployment Architecture

Kubernetes Cluster in Region A

Orderer Org

Peer Org

State DB
Peer
Chaincode

CA

Kafka
ZooKeeper
Orderer

Kubernetes Cluster in Region B

Peer Org

State DB
Peer
Chaincode

CA

Kubernetes Cluster in Region C

Peer Org

State DB
Peer
Chaincode

CA

Blockchain Application

Kubernetes Cluster

Blockchain Application

Kubernetes Cluster

Blockchain Application

Kubernetes Cluster
BaaS Packaging, Deployment and Service Orchestration

**Problem**

Complex system
Many containers, services and images
Strong service inter-dependency

**Solution**

Package & Deploy
- Helm Chart
- Container Image

Config
- ConfigMap, Secret
- Chart Values

Repository
- OSS
  - Container Registry Service

Service Orchestration
- Chart Template, Chart Hook
- Init Container, Shell Scripts
BaaS High Availability

Blockchain Application HA

Hyperledger Fabric Component HA
Orderer, Peer, Kafka, ZooKeeper, CA

Kubernetes Cluster HA
Master Nodes, Worker Nodes

BaaS Management Component HA

Cloud Infrastructure and Resource Service HA
Multi-AZ, ECS, network, storage, etc.

SLB
Service Discovery
Pod Anti-Affinity
BaaS Data Persistence

- Choosing storage type
  - File system storage: Alibaba Cloud NAS
  - Block storage: Alibaba Cloud Disk
  - Both provide reliability
    - NAS: 99.999999999%, Cloud Disk: 99.9999999%
  - Choosing SSD type for I/O performance

- Kubernetes Storage
  - PV + PVC

- Why Alibaba Cloud NAS is fit for blockchain ledger
  - Seamless and dynamic storage expansion, without interrupting/restarting applications
  - The larger volume, the larger iops performance

Problems with local disk:
- Scheduled to different nodes by Kubernetes
- If using NodeSelector, not able to cope with disaster recovery
BaaS Consortium Networking Challenge

Single VPC Consortium Network
Similar to private chain

Public Network-based Consortium Network
Not satisfying advanced security requirement

Leased Line-based Consortium Network
Complex and expensive
CEN-based Secure Consortium Networking Solution

- VPC
- Local IDC
- Blockchain Nodes
- DNS PrivateZone (Inter-VPC domain name resolution)
• Existing problems with chaincode container
  • Independent of Kubernetes world, making it difficult for lifecycle management against chaincode container
  • Not able to leverage Kubernetes namespace isolation, network policy to guarantee security of chaincode container
Smart Contract – Chaincode Container

• Solutions
  • Option 1: Having chaincode container in Kubernetes (like as pod)
    • Ideal solution, achieving uniform lifecycle management for all Fabric nodes, while being able to leverage Kubernetes network policy to control access
    • Hyperledger Fabric community requirement (JIRA), not implemented yet: https://jira.hyperledger.org/browse/FAB-7406

Future Outlook: Scheduling container container to serverless kubernetes, achieving security isolation between kernels
Smart Contract – Chaincode Container

- Solutions
  - Option 2: Running chaincode in Docker-in-Docker (DIND)
    - Sample configuration yaml
    - Observation and analysis

Advantages
- No need to rely on /var/run/docker.sock of host node
- No need to explicitly clean up chaincode images on each worker nodes

Limitations
- Slow in each creating or recovering peer nodes, since dind needs to pull fabric-ccenv image (1.4GB), while traditional way only needs to pull once on worker node
- Chaincode instantiate slows down a bit
- Slow in restarting peer nodes or whole Fabric network (reusing data directories) - same reason as #1
- In industry practice, DIND approach is mainly for CI/CD, but challenges (like stability) remain if applied to production
- Still not solving security access control and isolation problems of chaincode container
Smart Contract – Chaincode Container

- **Options**
  - **Option 3**: Comprehensive configurations to address key challenges
    - Fabric peer setting to ensure communication with chaincode
    - Using `docker rm` and `docker rmi` commands to clean up chaincode container and image (‘dev’ prefix)
    - **Candidate 1**: DaemonSet + lifecycle.preStop.exec.command (cleanup after delete)
    - **Candidate 2**: initContainer (cleanup before deploy)
    - Using iptables rules to restrict network access of chaincode container
    - Configure iptables rules of Kubernetes worker nodes during Helm Chart installation
    - Restrict chaincode's access to Kubernetes network and public network

```yaml
containers:
  - name: peer
    env:
      - name: CORE_VM_ENDPOINT
        value: "unix:///host/var/run/docker.sock"
      - name: CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE
        value: "bridge" # Use docker's host network
      - name: CORE_PEER_ADDRESS
        value: "true" # Ensure to get IP address of peer pod
      - name: CORE_PEER_ADDRESS
        value: "{{peerName}}" # Include the peer's pod IP
      - name: CORE_PEER_ADDRESS
        value: "true" # Include the orgName and peerName
      - name: CORE_PEER_ADDRESS
        value: "true" # Include the peerIndex

    volumeMounts:
      - name: docker-socket
        mountPath: /host/var/run/docker.sock
```

Auto resolve peer's pod IP and pass in while starting chaincode container.
Typical Pattern of Blockchain System on Kubernetes
Demo

- Fast building Cross-enterprise account, cross-region consortium
- Adding new organization and new channel dynamically
- Cross-enterprise collaboration (invite, approve)
- Risk control (SMS verification against key operations)
- Deploying chaincode and client SDK application (Marbles)
Question ?
Thank You