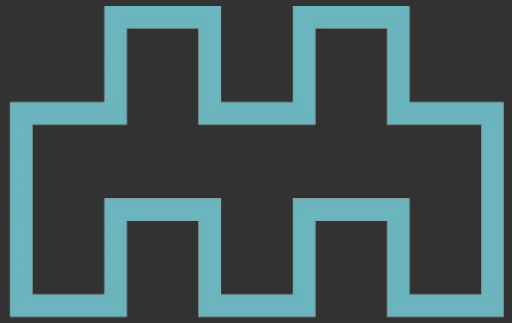


MINISTRY

BUILDING A SCALABLE MOBILE GAME BACKEND IN ELIXIR

Petri Kero
CTO / Ministry of Games

MOBILE GAME BACKEND

CHALLENGES

- Lots of concurrent users
- Complex interactions between players
- Persistent world with frequent state mutation
- Single unified game world

MOBILE GAME BACKEND

CHALLENGES

- Lots of concurrent users
- Complex interactions between players
- Persistent world with frequent state mutation
- Single unified game world

ELIXIR TO THE RESCUE

- Built-in clustering
- Distributed messaging
- Great for stateful servers

MOBILE GAME BACKEND

CHALLENGES

- Lots of concurrent users
- Complex interactions between players
- Persistent world with frequent state mutation
- Single unified game world

ELIXIR TO THE RESCUE

- Built-in clustering
- Distributed messaging
- Great for stateful servers

MORE BENEFITS

- Fault tolerant
- Rapid development with small team
- Tooling & documentation
- Learning curve

DEPLOYMENT OVERVIEW

CLUSTERING

- Running on Kubernetes
- Fully connected Erlang mesh
- Stateful servers keep hot state in memory

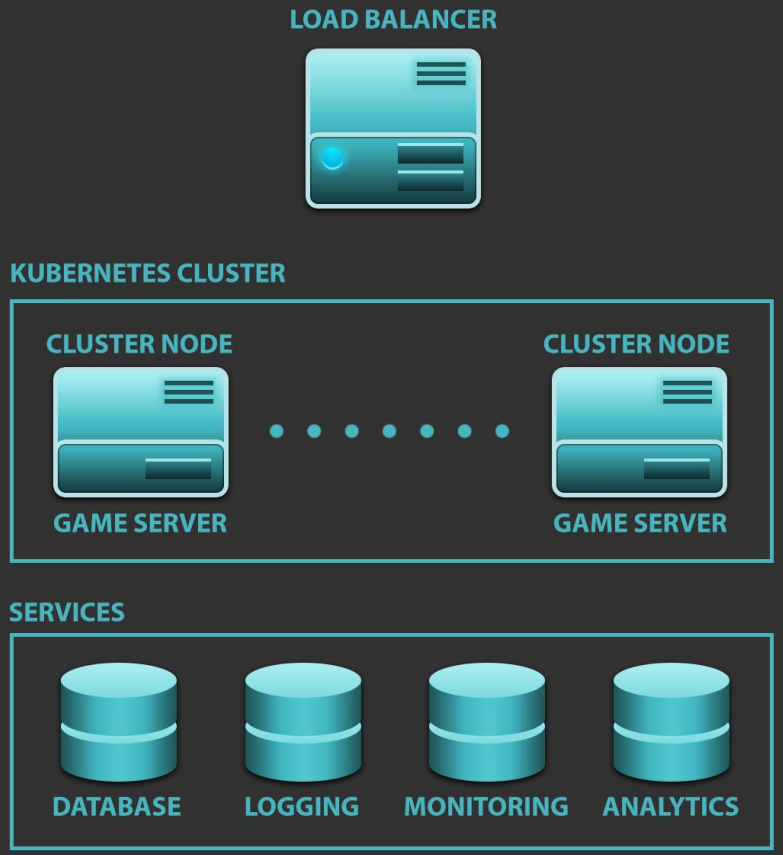
DEPLOYMENT OVERVIEW

CLUSTERING

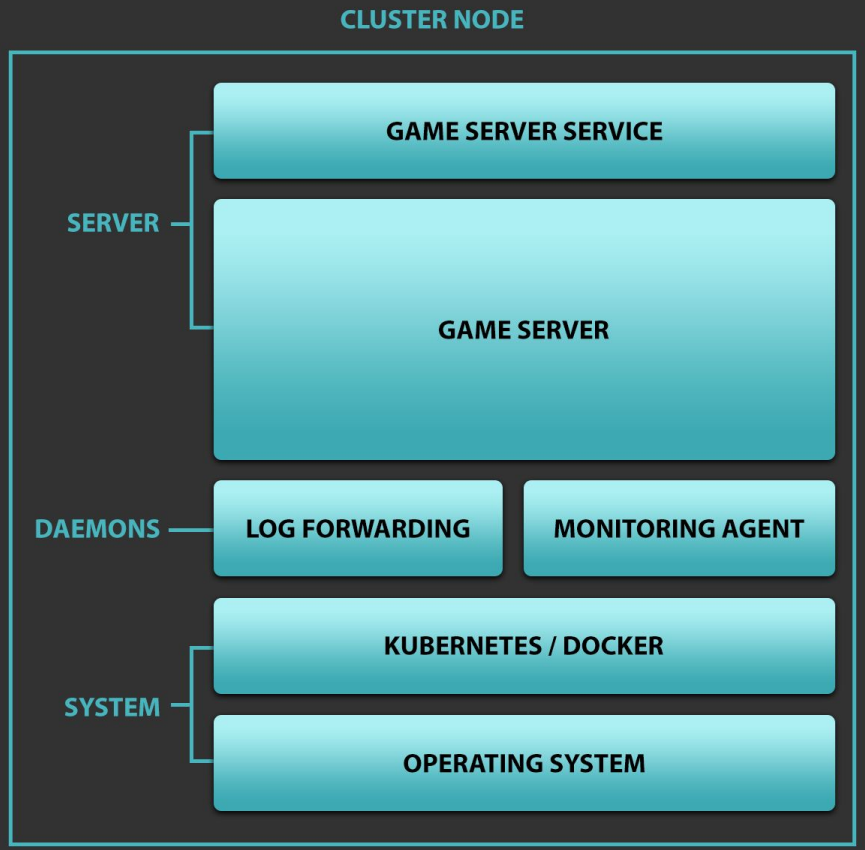
- Running on Kubernetes
- Fully connected Erlang mesh
- Stateful servers keep hot state in memory

EXTERNAL SERVICES

- Load balancer
- Key-value database
- Log storage/analysis
- Infrastructure monitoring
- Game analytics



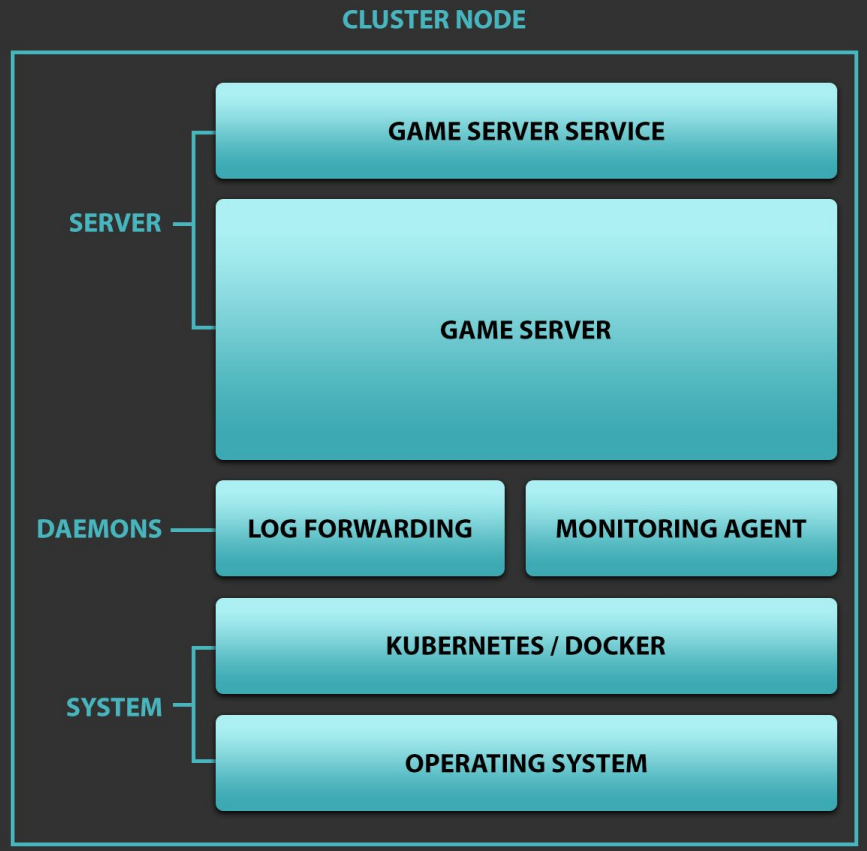
DEPLOYMENT DETAILS



DEPLOYMENT DETAILS

GAME SERVER

- One Distillery release
- Packaged as Docker container
- Kubernetes-orchestrated deployment
- Built & deployed using Jenkins pipelines



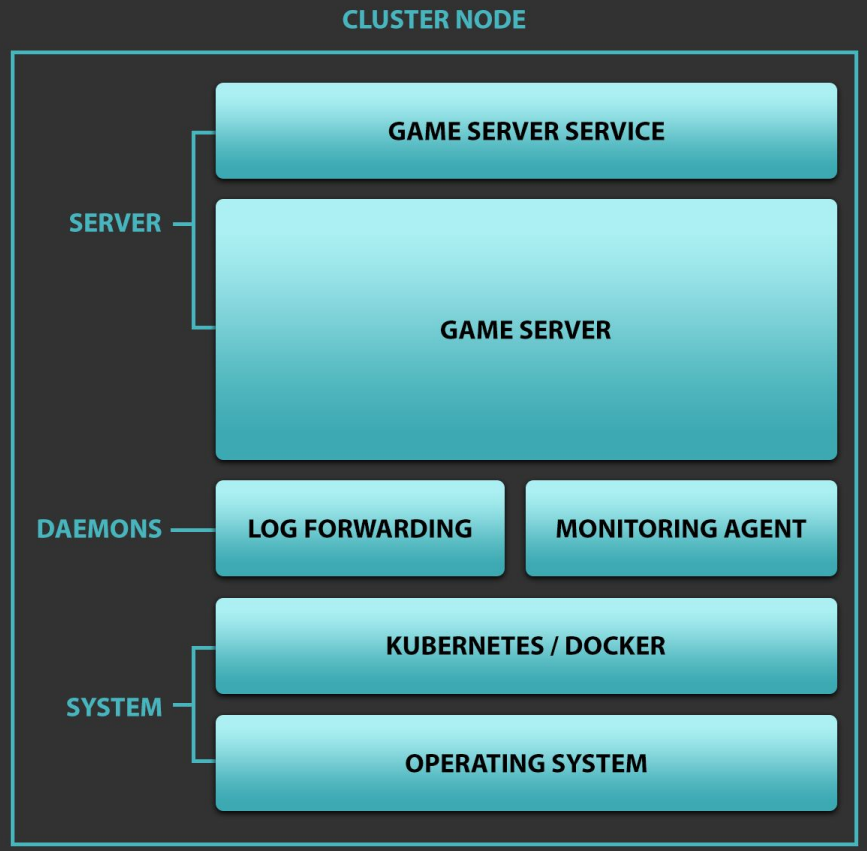
DEPLOYMENT DETAILS

GAME SERVER

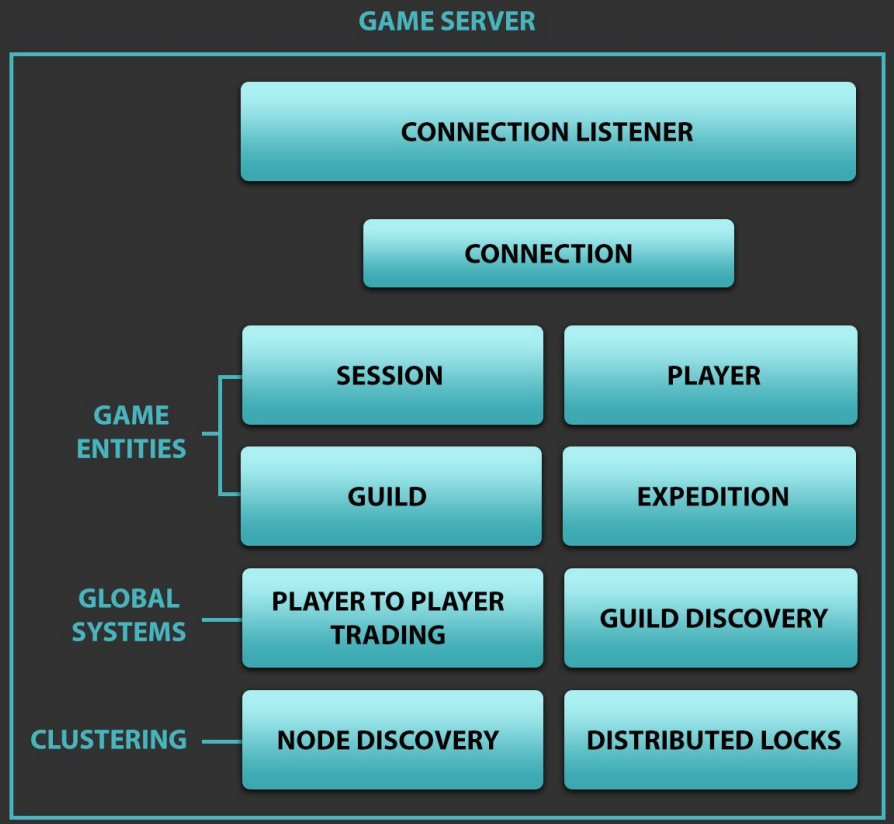
- One Distillery release
- Packaged as Docker container
- Kubernetes-orchestrated deployment
- Built & deployed using Jenkins pipelines

DAEMONS

- Logspout forwards logs to Papertrail
- DataDog monitoring agent



GAME SERVER COMPONENTS



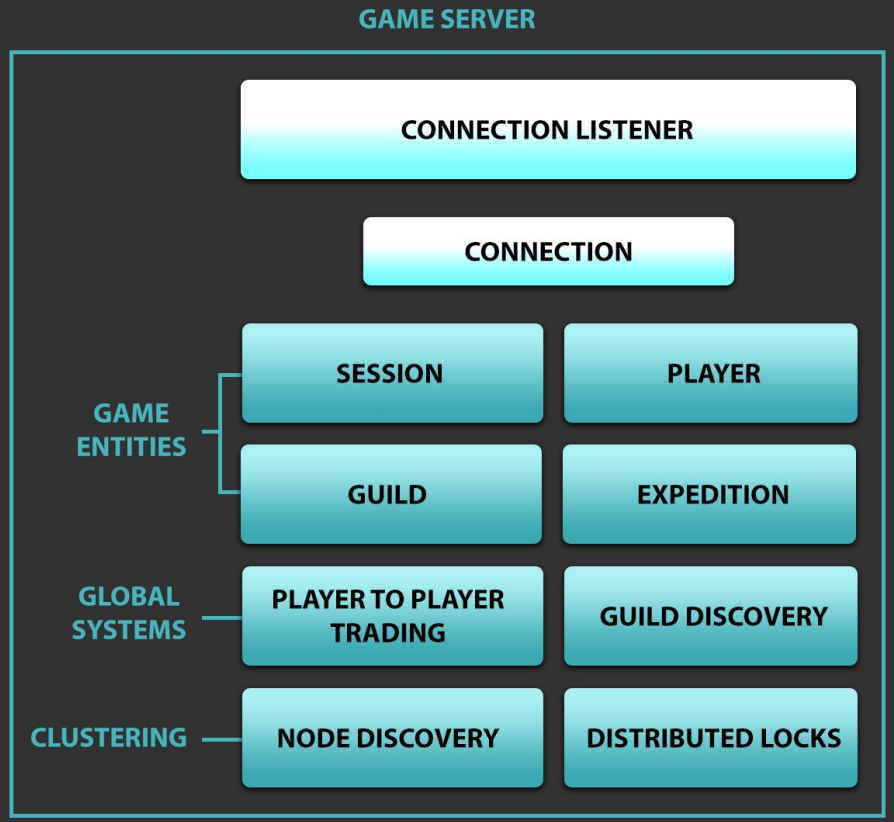
GAME SERVER COMPONENTS

CLIENT CONNECTIONS

- Uses ninenines/ranch
- Persistent TCP connections
- Bidirectional RPC-like API
- Spawns (or resumes) Session for Player

DISCONNECT HANDLING

- Session resuming for short disconnects
- Client mostly assumes success of requests
- Client restarts after longer timeout



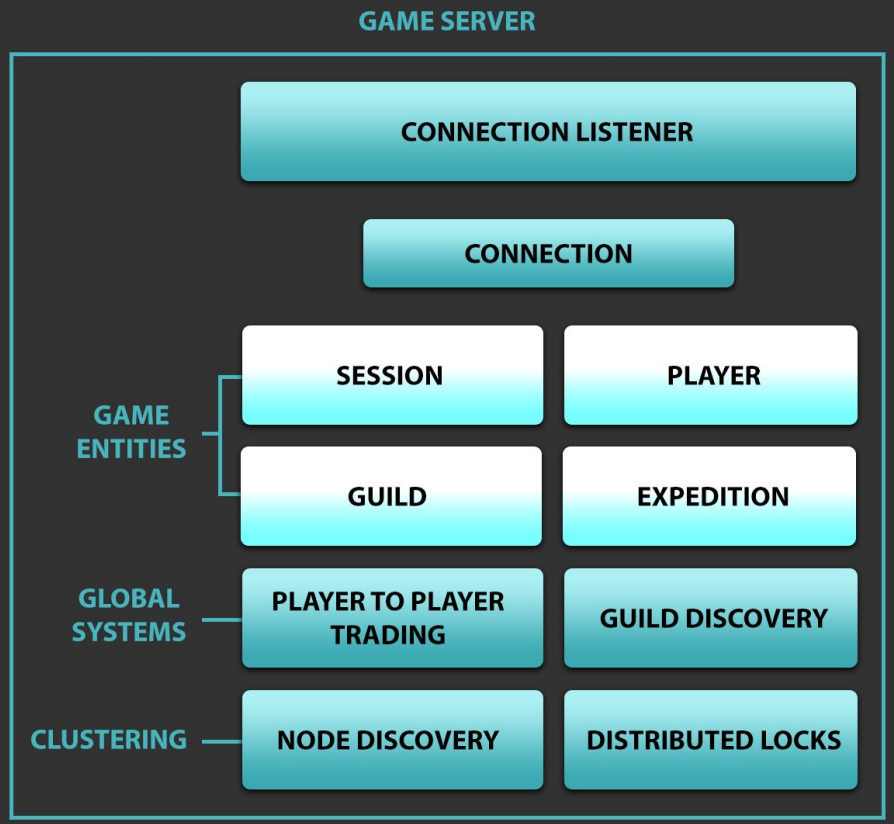
GAME SERVER COMPONENTS

SESSION
- Glue between connections and game world
- Implements transactions between Players

PLAYER
- Stores and updates player state
- Subscribes to member Guild for updates

GUILD
- Main routing place of helps/request between Players

EXPEDITION
- Weekly events for Guild vs. Guild competition



GENENTITY

EXTENDED GENSERVER

- Discovery with global process registry
- Lifecycle management
- Persisting into database
- PubSub-style communication
- Debugging & introspection

LIMITATIONS

- No guarantee of uniqueness
- Atomicity only within GenEntity

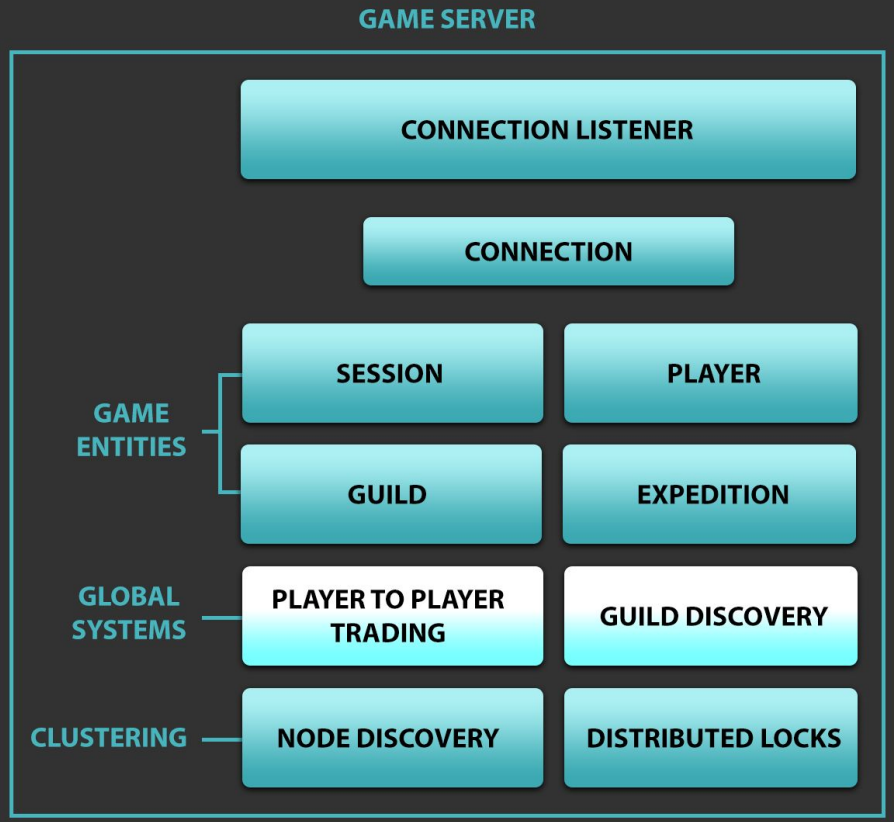
GAME SERVER COMPONENTS

PLAYER-TO-PLAYER TRADING

- Players publish items for sale into the system
- Online layers continuously receive item offers
- Full connectivity within cluster

GUILD DISCOVERY

- Keeps track of all guilds in the game
- Helps players find guilds to join



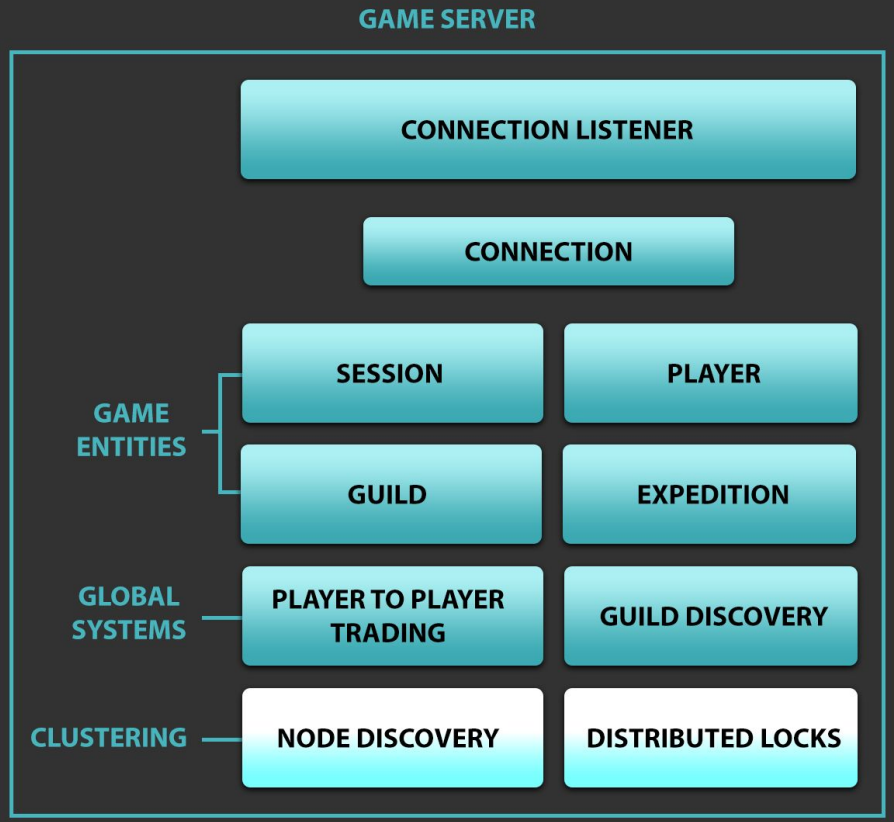
GAME SERVER COMPONENTS

NODE DISCOVERY

- Syncs Erlang cluster to Kubernetes state
- Simple layer on top of bitwalker/libcluster
- Uses Kubernetes services to discover peers

DISTRIBUTED LOCKS

- Global process registry for game entities
- Based on sloppy quorum model
- No consistency guarantees
- Distributed for high throughput



DISTRIBUTED LOCKS

OVERVIEW

- Grants timed leases to entities
- Sloppy quorum of three replicas
- Distributed version of wooga/locker

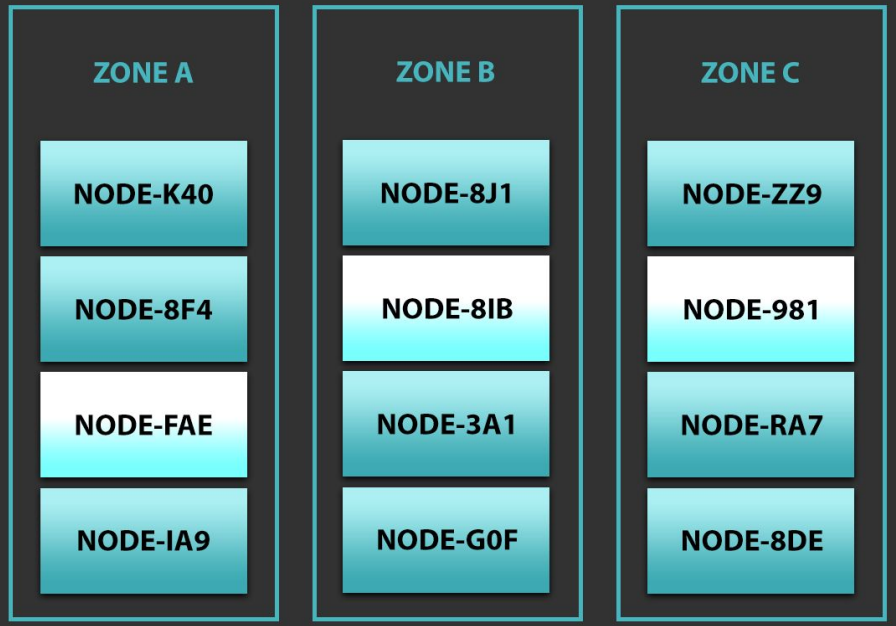
DISTRIBUTED LOCKS

OVERVIEW

- Grants timed leases to entities
- Sloppy quorum of three replicas
- Distributed version of wooga/locker

TOPOLOGY

- Static grid of vnodes
- Servers compete for ownership of vnodes
- Zone-aware distribution



DISTRIBUTED LOCKS

OVERVIEW

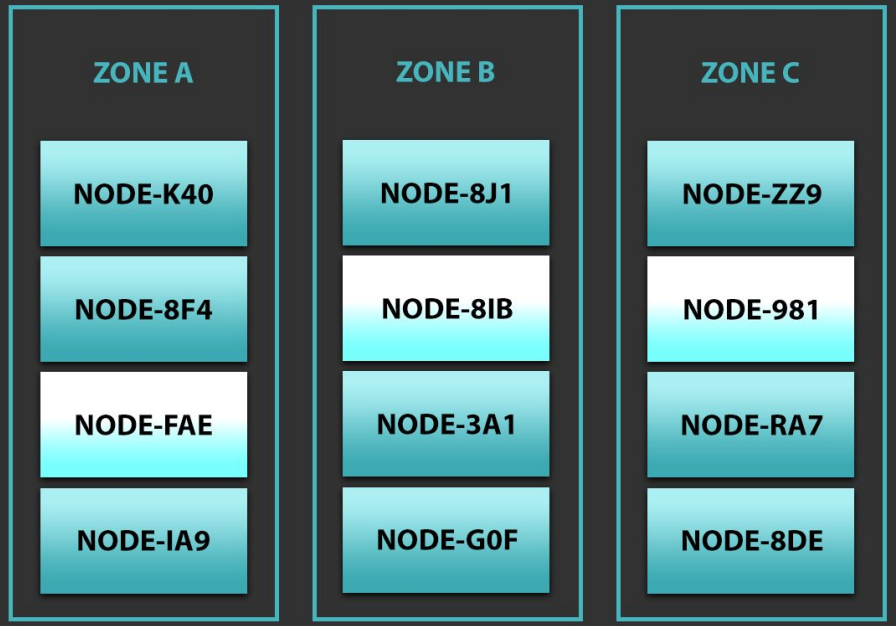
- Grants timed leases to entities
- Sloppy quorum of three replicas
- Distributed version of wooga/locker

TOPOLOGY

- Static grid of vnodes
- Servers compete for ownership of vnodes
- Zone-aware distribution

CONSISTENCY VS. AVAILABILITY

- Best-effort consistency, no guarantees
- Overwrite lost entities for higher availability
- Versioning for conflict resolution



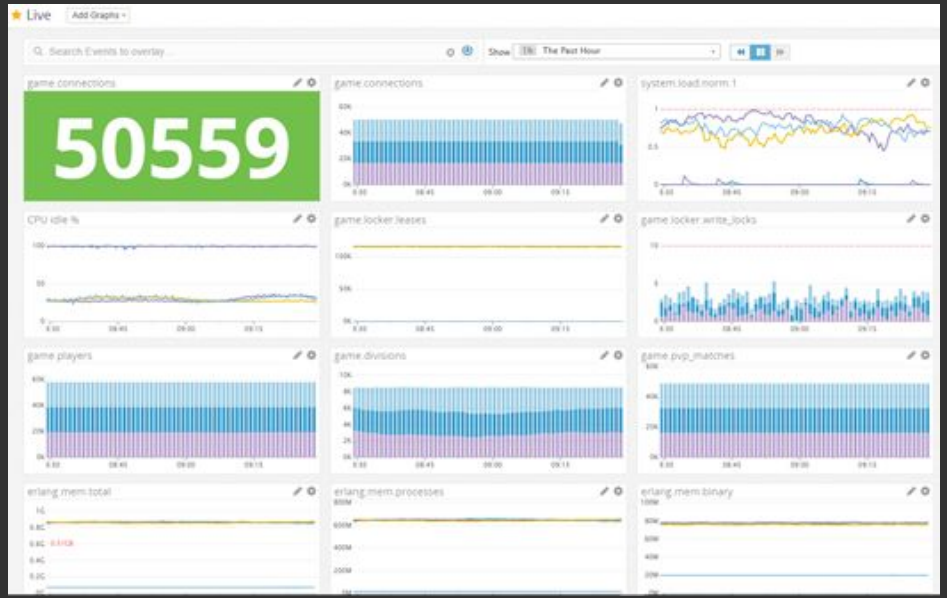
SCALING IT UP

SETUP

- Use AI clients to simulate traffic
- See how far system scales

BOTTLENECKS

- 1k: OS file limit for process
- 3k: Logger overwhelmed
- 4k: Ranch max connection setting
- 10k: DynamoDB provisioned bandwidth
- 20k: Test client spawn rate
- 50k: Disk full (after 12h)



SCALING IT UP

MORE BOTTLENECKS

- 51k: CPU limit on 3-node 8 vCPU system
- 150k: Locker overwhelmed
- 175k: DynamoDB client HTTPS overhead
- 420k: Client spawn rate

RESULTS

- 420k concurrents on 8-node 36 vCPU cluster
- 52k concurrents per node (1.4k per vCPU)
- 3GB memory used per node
- Expected to scale further

Nodes

Name	CPU	Memory used	Processes	Ports
spells@qa-game-server-0-us-east-1.ministryofgames.internal	39.8%	3.08 GB	157295	51842
spells@qa-game-server-3-us-east-1.ministryofgames.internal	41.4%	3.02 GB	155470	51377
spells@qa-game-server-6-us-east-1.ministryofgames.internal	41.0%	3.02 GB	155266	51549
spells@qa-game-server-4-us-east-1.ministryofgames.internal	41.6%	3.04 GB	155819	51533
spells@qa-game-server-1-us-east-1.ministryofgames.internal	42.0%	3.09 GB	156644	52008
spells@qa-game-server-7-us-east-1.ministryofgames.internal	41.1%	3.06 GB	155524	51710
spells@qa-game-server-5-us-east-1.ministryofgames.internal	40.5%	3.07 GB	156768	51915
spells@qa-game-server-2-us-east-1.ministryofgames.internal	43.7%	3.07 GB	156872	51783

Petri Kero
CTO / Ministry of Games
petri@ministryofgames.io