Stateless vs Stateful Service

- Stateless
  - No Persistent Storage
  - Mortal
  - Scaling can be done independently
  - Client side cookies can be used to make stateless service
- Stateful
  - Stable, unique network identifiers.
  - Stable, persistent storage.
  - Ordered, graceful deployment and scaling.
  - Ordered, graceful deletion and termination.
  - Ordered, automated rolling updates.

"Stateful" Container	"Stateless" Container
• **Secrets** - public/private keys, password, etc • **Databases** - databases, sharded, clustered. • **Logs** - to collect support bundles, run analytics for data mining, etc. • **Other** - CI repo data, transcoded bits...	• Nothing to Disk • Web Front-End • Can stop and start as many containers as you like • Like http is stateless • Container is ephemeral • Does not care about what has happened or changed.

Stateful or Stateless Design

Stateful

Server maintains client-specific state

- Shorter requests
- Better performance in processing requests
- Cache coherence possible
  - Server can know who's accessing what
- File locking possible

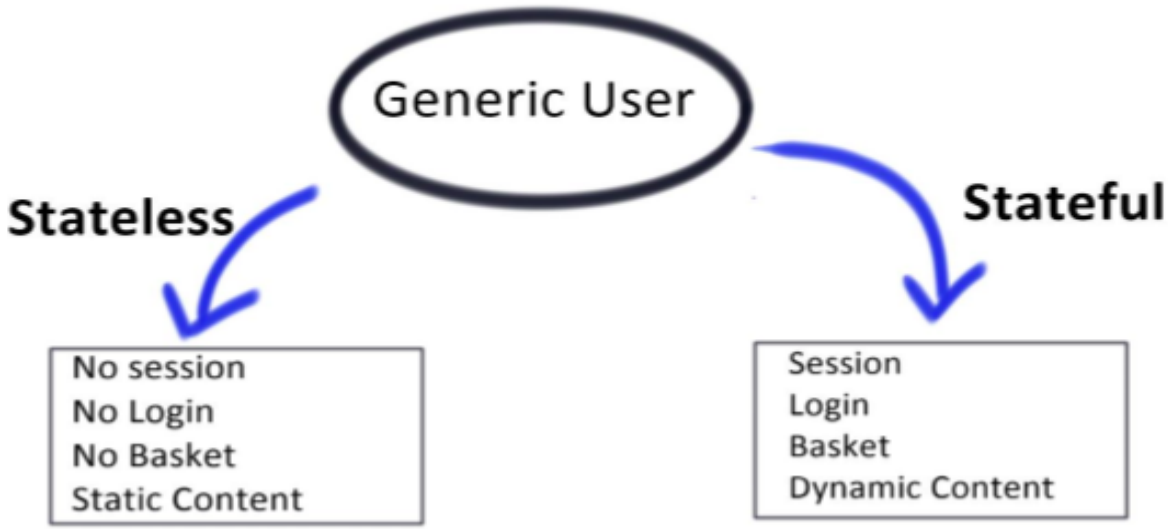
Stateless

Server maintains no information on client accesses

- Each request must identify file and offsets
- Server can crash and recover
  - No state to lose
- No open/close needed
  - They only establish state
- No server space used for state
  - Don't worry about supporting many clients
- Problems if file is deleted on server
- File locking not possible

Stateful Application Deployment

- Some application need “state” for keep application flow and store some information on “session” (Such as login’s session id) that store on web server or middle tier server module
- Ex: Joomla, Wordpress, Mantis (Bug Tracking), Normal etc



StatefulApplication Deployment

- Kubernetes: Production Workload Orchestration
- Considering o Original “HTTP” protocol is “stateless” o Stateful application need to keep session by web/app server and keep “cookies” on client for pass authentication o Work on memory for keep session (Fast/Easy but consume resource) o Many problem with native mobile app/Centralize Problem o Scale will effect for consideration traffic redirect to correct server (Keep state)
- Awareness o Container is naturally design for “stateless” application o All load-balance/dispatch job is not aware about “state” of application inside

StatefulApplication Deployment

Kubernetes: Production Workload Orchestration Solution ?

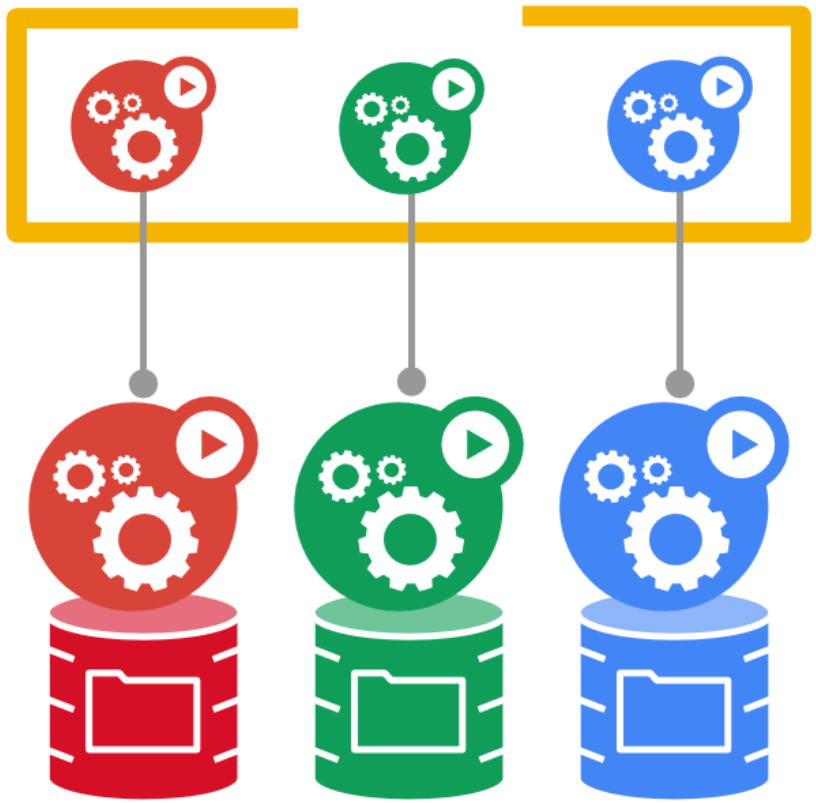
- SDS (Software-Defined Storage) for make centralize storage pool
- Share centralize storage pool for all node
- For Web/App Server
- Keep application path / Session path on storage pool
- Every server will read/write on same place
- For Database Server Many option for operate (depend on type of database)
- Active/Active
- Active/Hot-Passive
- Active/Cold-Passive
- Postgres Kubedb Tool (Beta)
- Idea also keep data on storage pool

Goal: enable clustered software on Kubernetes

- mysql, redis, zookeeper, ...

Clustered apps need “identity” and sequencing guarantees

- stable hostname, available in DNS
- an ordinal index
- stable storage: linked to the ordinal & hostname
- discovery of peers for quorum
- startup/teardown ordering



Demo