BURNING DOWN THE HOUSE
BACKUP AND RESTORE OF CLOUD FOUNDRY

Therese Stowell
Pivotal
@midatlanticitez

Henry Stanley
Pivotal
@henryaj
WHAT WE’RE GOING TO TALK ABOUT

- Current approaches
- Issues
- Our solution
- The future
WHY BACKUP AND RESTORE CF?

- It's the first thing operators ask!
- Mission-critical apps run on CF - every company is a software company
- Operators need to be able to recover CF, apps, data services
- Dev workflows use CF too
WHAT CAN GO WRONG?

- Data corruption
  - a failed upgrade
  - user error (GitLab melts down after wrong directory deleted, backups failed)
  - security issues, e.g. ransomware!
- Hardware failure
  - SAN failure
  - datacenter failure*

*burning down the house
SO WHAT DOES IT MEAN TO BACK UP CLOUD FOUNDRY?
WHERE DOES THE STATE IN CLOUD FOUNDRY LIVE?
Cloud Controller

cf push

Cloud Controller

Diego

insert metadata

save source

DB

metadata

Blobstore

buildpacks

source

😊
Cloud Controller

Diego

retire buildpack + app source

stage app

cf push

😊

metadata

DB

blobstore

buildpacks

source
Cloud Controller

Diego

cf push

save droplet metadata

metadata

save droplet

buildpacks

source

droplet

DB

blobstore

😊
WHAT ABOUT CF ITSELF?

BLOBS + DB
WE NEED TO GO DEEPER
upload release

save metadata

save release

DB

blobstore

release

release
 telefone

DB blobstore

BOSH

save deployment manifest

DB

release

manifest

blobstore

release
WHAT ARE CURRENT APPROACHES?
TOOLS THAT REACH UNDER THE HOOD

- Reach into CF components and back up their state

Issues

- Fragility
- Knowledge of CF
- Backwards compatibility
- Consistency
REPLICAS

- Active-Active
- Active-Passive

Issues

- Really Difficult™ to get right
- Doubles your cost
- Does not mitigate against malware because both instances have the issue
SNAPSHOTS

IaaS snapshotting

Issues

- Pulling data underneath
- Not all state is needed
- Slow for some IaaSes
- Consistency across components
RECREATE EVERYTHING

- Script CF configuration
- Recreate CF via pipelines
- Repush apps via pipelines

Issues

- Not all teams have pipelines
- Requires discipline and cross-org collaboration
- Time to recovery is long
- Service instance data not recoverable
OTHER COMMON THEMES

- Artifact Management
  - Encryption
- Scheduling
- Retention
- Partial or Complete
- Incremental
PROBLEM SPACE

- Data Consistency
- Service Unavailability
- How to Sync Data to Disk
- Encryption on the Wire
- How to Restore
- Retention Policies
- Scheduling
- Snapshots
- Integration with Archives
- What Data is Relevant for Backups
- Role Based Access Control
- Which Components Need Backing Up
- Version Compatibility
- Cross Deployment Backups
- Encryption at Rest
BBR: A NEW MODEL
PROBLEM SPACE

- Data Consistency
- Service Unavailability
- How to Sync Data to Disk
- Encryption on the Wire
- How to Restore
- Retention Policies
- Scheduling
- Snapshots
- Integration with Archives
- What Data is Relevant for Backups
- Role Based Access Control
- Which Components Need Backing Up
- Version Compatibility
- Cross Deployment Backups
- Encryption at Rest
HOW TO ORCHESTRATE

- DATA CONSISTENCY
- ENCRYPTION ON THE WIRE
- RETENTION POLICIES
- SCHEDULING
- ROLE BASED ACCESS CONTROL
- ENCRYPTION AT REST
- INTEGRATION WITH ARCHIVES
- CROSS DEPLOYMENT BACKUPS

HOW TO BACKUP & RESTORE

- WHAT DATA IS RELEVANT FOR BACKUPS
- WHICH COMPONENTS NEED BACKING UP
- HOW TO RESTORE
- VERSION COMPATIBILITY
- SERVICE UNAVAILABILITY
- HOW TO SYNC DATA TO DISK
- SNAPSHOTS
**Orchestrator**

- Data Consistency
- Encryption on the Wire
- Retention Policies
- Scheduling
- Role Based Access Control
- Encryption at Rest
- Integration with Archives
- Cross Deployment Backups

**Component**

- What Data is Relevant for Backups
- Which Components Need Backing Up
- How to Restore
- Version Compatibility
- Service Unavailability
- How to Sync Data to Disk
- Snapshots
CONTRACT BETWEEN ORCHESTRATOR AND COMPONENT

Backup / Restore
Orchestrator

Component to be backed up / restored
THE ORCHESTRATOR

For a backup, the orchestrator must trigger hooks in the component in this order:

1. Lock
2. Backup
3. Unlock

For a restore, the orchestrator must trigger hooks in the component in this order:

1. Lock
2. Restore
3. Unlock
THE ORCHESTRATOR

For a backup, the orchestrator must trigger hooks in the component in this order:

1. Lock
2. Backup
3. Unlock

For a restore, the orchestrator must trigger hooks in the component in this order:

1. Lock
2. Restore
3. Unlock
4. Orchestrator responsible for moving artifacts from/to the component
THE COMPONENT

Component must provide hooks for backup and restore

- Lock / Backup / Unlock
- Lock / Restore / Unlock
THE COMPONENT

Component must provide hooks for backup and restore

- Lock / Backup / Unlock
- Lock / Restore / Unlock

Component can implement any / all / none of the hooks, depending on what is needed
THE COMPONENT

- Lock / Backup / Unlock
- Lock / Restore / Unlock
- Component must put generated backup artifacts in particular location
- Component must take backup artifact from a particular location for restore
THE COMPONENT

- Lock / Backup / Unlock
- Lock / Restore / Unlock
- Component must put generated backup artifacts in particular location
- Component must take backup artifact from a particular location for restore
- Component can specify order of locking (optional)
THE CONTRACT: ALL THE THINGS

Orchestrator must trigger hooks in this order for a backup:

1. Lock
2. Backup
3. Unlock

Orchestrator must trigger hooks in this order for a restore:

1. Lock
2. Restore
3. Unlock

Orchestrator responsible for moving artifacts to/from component

Component must provide hooks for backup and restore

Component can implement any / all / none of the hooks, depending on what is needed

Component must put backup artifacts in particular location

Component must take artifacts for restore from a particular location
Knowledge of how to back up and restore a component is encapsulated in the component

- Responsibility for writing and maintaining scripts sits with the component author
- Backup and restore logic can change as component changes
- Component authors are best placed to write the hooks
- Addresses fragility and compatibility issues
CONTRACT BENEFITS

- Back up is triggered at a logical level rather than a hardware level
  - Data can be flushed
  - Locking enables consistency across data stores
  - Backup / restore can be smart
    - A backup script can choose not to back up all the data, alter the data in some form, add metadata, encrypt
    - A restore script can regenerate credentials, decrypt, alter the data
  - Data transfer decoupled from backup enables less downtime
CONTRACT BETWEEN ORCHESTRATOR AND COMPONENT

Backup / Restore Orchestrator

Component to be backed up / restored
OUR IMPLEMENTATION OF THE CONTRACT: BOSH BACKUP AND RESTORE FRAMEWORK

- Orchestrator is the BOSH Backup and Restore (BBR) binary
- Component is a BOSH job
- Unit of backup is BOSH deployment
- Why?
  - all components in CF are BOSH deployments
  - the BOSH director is itself a BOSH release
  - for an operator, a BOSH deployment is a logical unit of backup
THE BBR BINARY

- A CLI that runs on a jump box
  pre-backup-check
  backup
  backup-cleanup
  restore
  restore-cleanup

- Can backup / restore BOSH deployments and BOSH directors

- BBR backs up CF state and software, but restore doesn’t deploy the software
  - Restore will put the data back, not install the software, like mysql restore.

- Open sourced: Cloud Foundry Extensions Incubator!
Backup → BBR

Cloud Controller → gorouter → Diego

Lock

DB • UAA DB • blobstore
Backup
BBR
Cloud Controller
gorouter
Diego
Backup
Backup
Back up
routing
table
DB
UAA DB
blobstore
Backup → BBR → Drain

Cloud Controller  → gorouter  → Diego

DB  → UAA DB  → blobstore
DEMO!

BACK UP ELASTIC RUNTIME
DEMO! BACK UP ELASTIC RUNTIME
RELEASES THAT SUPPORT BACKUP AND RESTORE VIA BBR

- BOSH Director
- Credhub
- UAA
- Elastic Runtime (Pivotal CF)
- Coming soon:
  - CF
  - Data Services
BBR CLI IS A SMALL SHARP TOOL

THE BACKUP AND RESTORE ECO SYSTEM

- BBR CLI doesn’t do:
  - encryption
  - scheduling
  - artifact management
  - role-based access control
  - moving artifacts to secondary backup store
Deduplicate and secure BBR backup artifacts

Use BoostFS to store artifacts directly on Data Domain

- Backup artifact protection storage with 10-30x data reduction

Initial Testing

- 75% reduction in backup storage requirements
- 20% reduction in backup transfer time
- Reduced network bandwidth (due to deduplication)

Links:
https://www.emc.com/collateral/white-papers/h16650-v1.0.pdf
https://youtu.be/1wCtptcQZZI
POTENTIAL CONTRACT EXTENSIONS

FUTURE OF BBR CONTRACT

- External data stores
- Backup artifact validation
- Partial backup and restore
- Incremental backup and restore
FUTURE OF BBR CLI

‣ Use a BOSH agent to drive contract rather than SSH

‣ Optimizations
COMING SOON

B+R MORE THINGS

- Data services
- External databases
- External blobstores
Making CF more backup-able

- Enabling read-only mode during backup
  - Releases remain available for reads, minimizing downtime
- Minimizing data references across data stores
- Event streams
  - Replayable series of events to recreate state
TO SUMMARIZE

- Cloud Foundry Backup: State + CF
- Current Approaches & Issues
- The Contract
- Our Implementation: BOSH Backup & Restore
- The Future
WHERE TO FIND BBR

- [https://github.com/cloudfoundry-incubator/disaster-recovery-acceptance-tests](https://github.com/cloudfoundry-incubator/disaster-recovery-acceptance-tests)
- [http://docs.cloudfoundry.org/bbr/index.html](http://docs.cloudfoundry.org/bbr/index.html)
- [https://cloudfoundry.slack.com #bbr](https://cloudfoundry.slack.com #bbr)
THANKS!

QUESTIONS?

@midatlanticitez  @henryaj