The Anti-Pattern Wall of 2011 has Crumbled: Stateful Applications in Containers

Open Source Summit North America
September 2017
Speaker

Steve Wong
Strategic Open Source Partner Engineer
{code} Dell Technologies

Community participant:
- Container Storage Interface
- DC/OS
- Kubernetes
- Mesos
- REX-Ray
Agenda

1. History of containers & the 12 factors
2. Container Orchestrators vs. Paas
3. Orchestrator Support for Stateful Services
4. How and why you should get involved with stateful now
Container History Lesson

- 1979: chroot Unix V7
- 1981: BSD chroot
- 2000: FreeBSD jails
Later: growing adoption, by the cognoscenti
Advent of the PaaS platforms

- 2009: Heroku
- 2011: Cloud Foundry
- 2011: dotCloud
12 factors published

- 2009: Heroku
- 2011: Cloud Foundry
- 2011: dotCloud

The 12 factors published in 2011.
What are the 12 factors?

A collection of patterns and anti patterns intended to guide designs for the Heroku Platform (PaaS)

A PaaS simplifies application deployment by providing, managing and abstracting base of foundational services, with compute and storage infrastructure. Cloud Foundry is another popular PaaS generally utilized with 12 factor principles
Docker and container orchestrator history

- 2009: Heroku
- 2010: Cloud Foundry
- 2011: dotCloud
- 2011: The 12 factors published
A PaaS platform takes more responsibility – often making decisions for you

A container orchestrator is less opinionated
What does the 12 factors say?

VI. Processes
Execute the app as one or more stateless processes

Why?
Easy to replace, upgrade, automate scale-up and scale-down
Don’t *useful* apps require state retention?

"Stateless" is a hoax—will only delegate the prob & make it harder to ctrl in terms of data integ, scale & availab

**There is no such thing as a "stateless" architecture. It's just someone else’s problem.**
Indeed, it is somebody else’s problem, and the 12 factors explicitly says this:

Twelve factor processes are stateless and share-nothing.

Any data that needs to persist must be stored in a stateful backing service.
What exactly is this *stateful backing service*?

In the era when the twelve factors was written it was typically a database.

And the twelve factors goes on to advise that it should be consumed via an API such as an https network service.
Stateful Backing Services are not just databases
What if it isn’t “the other guys problem”? Suppose you are that guy maintaining the “backing store”

There are valid reasons to do this yourself

• You want to pick your own tool and version
• You want to customize
• You want to stay portable across clouds
• You want to avoid database monoliths
Monolithic = bad
Container advantages make sense for stateful too

Container attributes:
• Consistent environment – same anywhere
• Dependency management - packaging

Orchestration can add:
• Health monitoring
• Automated rollouts and rollbacks
• Declarative configuration
• App/package store deploy experience
The 12 factors was and is still a great tool… just needs an update

- Still applies when using a PaaS
- Principle of using your stateful DB, NoSQL, etc. behind an API abstraction still applies

But it’s not
- A religious codicil
- A lawbook

It is something to be respected and considered when and where it makes sense
Stateful in a container is mainstream
Orchestrators have features specifically for stateful app support

<table>
<thead>
<tr>
<th></th>
<th>DC/OS</th>
<th>Kubernetes</th>
<th>Mesos</th>
<th>Swarm</th>
</tr>
</thead>
<tbody>
<tr>
<td>External persistent volumes</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Frameworks</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Packages</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Helm charts</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>operators</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>StatefulSets</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
local storage for state?

- **Availability Risk**
  - Migrate container to another host – your storage is gone
  - Host goes down – your service goes down

- **Scale Limitation**
  - Need more storage than the host has? Sorry…
How External Persistent Volume Mounts work

DIG THREE GRAVES UP THERE.

I'LL EXPLAIN IT TO YOU LATER.
How External Persistent Volume Mounts work
Mesos, DC/OS: Frameworks

- Fine grained control over what gets run, where
- Clustered App: control node startup order, scaling, placement
- Monitor health, manage backups, upgrades
DC/OS: Packages

- Manage and deploy applications from controlled curated repository(s)
- provide app store experience
  - UI
  - CLI
Kubernetes Helm Chart

- App store experience
- Update, Rollback
Kubernetes StatefulSet

- Stable, unique network identifiers.
- Stable, persistent storage.
- Ordered, graceful deployment and scaling.
- Ordered, graceful deletion and termination.
- Ordered, automated rolling updates.
Kubernetes Operators

- an application-specific controller that extends the Kubernetes API
- includes domain or application-specific knowledge to automate common tasks (create, configure, update, scale, etc) and manage instances of complex stateful applications.
Demo

OSS: Tutorial - Introduction to Stateful Applications on Kubernetes
• Tuesday 10:55am

MesosCon: Building your First Stateful DC/OS Service
• Thursday 4pm
In 2012, early cloud pioneer Heroku developed the Twelve Factor App, a set of rules and guidelines for helping organizations build cloud-native applications. It served as an excellent starting point, but as technology inevitably marched forward, some areas needed revisiting. To accommodate current best practices, this practical book expands on the original guidelines to help you build applications that not only function in the cloud, but also thrive.
What needs improvement in the stateful story

- Backup is ripe for standardization
- Storage “plugin drivers” are not standardized across platforms
- Replication, Volume resizing has “rough edges”
Efforts are underway in the community to address these – but no matter who you are, we could use your help.
Technology adoption is non linear

- 1870 First gasoline engine, attached to pushcart
- 1885 Karl Benz first gas automobiles offered for sale – hand made
- 1903 Ford model T mass produced
NYC 1900 – spot the car
NYC 1913 – spot the horse

43 years elapse

Accessibility to Mass Audience = big impact
Thank You
## References

<table>
<thead>
<tr>
<th></th>
<th>Docker Image</th>
<th>DC/OS package</th>
<th>Helm chart</th>
<th>K8s operator</th>
<th>K8s StaefulSet</th>
<th>Mesos Framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cassandra</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Elastic</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Hadoop</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Kafka</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>mongoDB</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>MySQL</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Redis</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
References

• Free O’Reilly Book, Beyond the Twelve Factor App:
  • http://www.oreilly.com/webops-perf/free/beyond-the-twelve-factor-app.csp

• This deck:
  • https://www.slideshare.net/SteveWong14/open-source-summit-northamerica-2017-stateful-apps-in-containers
Contacts

Steve Wong
@cantbewong

thecodeteam.com
thecodeteam.com/community/
@codeDellEMC