Item 299
git_comments:

1. Check if likely-looking bundle already installed to OSGi subsystem, but brooklyn not aware of it. This will often happen on a karaf restart where bundle was cached by karaf, so we need to allow it; can also happen if brooklyn.libraries references an existing bundle. If we're not certain that the bundle is identical
2. search for already-installed bundles.
3. e.g. Checksum would be missing if we brought under management a pre-installed bundle with an unusable url.
4. Identical bundle (by osgi location or binary content) already installed; just bring that under management. This will often happen on a karaf restart: bundles from persisted state match those cached by karaf,
5. Check if exactly this bundle is already installed
6. We were explicitly asked to bring an existing OSGi bundle under management; no equivalence check required
7. Uninstall and re-install the bundle. This is a good idea for brooklyn managed bundles that were in the karaf cache (when we can't determine that they are definitely identical). It's less good for pre-installed bundles, but if the user has said to deploy it or has referenced it in `brooklyn.libraries` then we'll go for it anyway! Let's hope they didn't reference `org.apache.brooklyn.core` or some such. We are this extreme because we want rebind to always work! If a user did a `force` install of a bundle, then we want to do the same on rebind (rather than risk failing). Instead of uninstall, we could update the bundle. Note however either way we won't be able to rollback if there is a failure
8. Specifically, we treat as "managed bundles" (to extract their catalog.bom) the contents of: - org.apache.brooklyn.core - org.apache.brooklyn.policy - org.apache.brooklyn.test-framework - org.apache.brooklyn.software-* - org.apache.brooklyn.library-catalog - org.apache.brooklyn.karaf-init (not sure why this one could end up in persisted state!)
9. No such Brooklyn-managed bundle.
10. e.g. happens if pre-installed bundle is brought under management, and then add it again via a mvn-style url. We wouldn't know the checksum from the pre-installed bundle.
11. e.g. repeatedly installing the same bundle
12. * * Gets the (internal) value to be used as the location in bundleContext.install(location). * It thus allows us to tell if a cached OSGi bundle is the same as a bundle we are about to * install (e.g. one we get from persisted state), or have retrieved from the initial catalog. * * Care should be taken to set the checksum before using the OSGi unique url.
13. Launch brooklyn, with initial catalog pointing at bundle that points at system bundle. Should bring it under brooklyn-management (without re-installing it).
14. Launch brooklyn again (because will have persisted the pre-installed bundle)
15. Launch brooklyn again (because will have persisted both those bundles)
16. * * Whether we reuse OSGi Framework depends if we want it to feel like rebinding on a new machine * (so no cached bundles), or a local restart. We can also use {@code reuseOsgi = true} to emulate * system bundles (by pre-installing them into the reused framework at the start of the test).
17. Launch brooklyn, with initial catalog pointing at system bundle. Should bring it under brooklyn-management (without re-installing it).
18. Launch brooklyn, and explicitly install pre-existing bundle. Should bring it under brooklyn-management (should not re-install it).
19. Aled thought we supported version ranges in 'brooklyn.libraries', but doesn't work here.

git_commits:

1. **summary:** PR #867: incorporate review comments
   message: PR #867: incorporate review comments And better handle pre-existing bundles.

github_issues:

github_issues_comments:

github_pulls:

1. **title:** BROOKLYN-546: fix rebind of system bundles
   body:

github_pulls_comments:

1. **body:** @ahgittin Still requires some more testing, but wanted to share this now for feedback. ---- As discussed with @ahgittin, there was an additional rebind issue when loading the initial catalog (and subsequently loading persisted bundles). Karaf will have cached OSGi bundles for that initial catalog. The initial catalog would duplicate some of these - it couldn't tell that the karaf bundle installed on the previous brooklyn run was identical to the bundle it wanted to install (it can't yet see the persisted brooklyn-managed records, and it can't understand osgi location `brooklyn:ID`). Possible solutions for that included: 1) make `brooklyn:ID` osgi location be a checksum (so the same osgi location would be used for two identical brooklyn-managed bundles). 2) clear the karaf cache of brooklyn-managed bundles (e.g. all those with location in the form `brooklyn:ID`), in the knowledge that the initial catalog and persisted state will reinstall them. 3) load initial catalog _after_ persistence (but we expect the initial catalog to tell us whether to remove some items from persistence so this isn't a good option without major re-work). 4) if we're not certain that it's identical, then uninstall and reinstall the bundle (which is what happens currently). This might be a problem if it's a core brooklyn module, but those should have `mvn:` urls so it _can_ confirm whether it's a duplicate). 5) rework rebind / initial catalog load so that we load placeholders for bundles that have been persisted (enough to do the deduping or install things referenced in initial catalog load). We opted for doing (1) and (4). --- Another issue (FAO @ahgittin) is that every time Brooklyn was started, we'd add to the persisted state another copy of each core brooklyn bundle! This is because the initial

catalog references things like `org.apache.brooklyn.policy`. The `ManagedBundle` we create to represent it gets an auto-generated unique id. We see that the bundle is already installed, so we don't re-install it in karaf (good). But we do still persist it to the `/bundles` persisted state sub-directory. I've worked around that by black-listing `org.apache.brooklyn.*` from being persisted (see `OsgiArchiveInstaller.isBlacklistedForPersistence(ManagedBundle)`. The assumption is that those will always ship with Brooklyn - we should use just the version that is shipped with the given release of Brooklyn. --- Another issue (similar, not yet fixed) is that we persist a new identical copy of `brooklyn-default-karaf-catalog` every time we restart Brooklyn. This is because the `ManagedBundle` for the initial catalog has an auto-generated unique id. As above, we do not install the duplicate bundle, but we still persist it. Luckily rebind is not as bad as you might think! It spots that an identical bundle is already installed, so doesn't install it or parse the catalog.bom of every copy that is in persisted state. Possible solutions for this include: 1) changing the persistence file/object names to use the `symbolicName_version.jar` format (similar naming to the old `/catalog` directory). 2) only persist the bundle if we actually installed it (i.e. it wasn't a duplicate). 3) search persisted state for a duplicate, and don't add if it already exists. Option (1) is very tempting, given that is it's logical unique identity: we shouldn't be persisting (and thus installing) multiple things with the same name:version. We'd need to be careful with backwards compatibility if someone has `/bundles` that uses the unique ids. We could later add additional checks to only replace the persisted file if its checksum had actually changed. Option (2) feels wrong - what is persisted depends on what is in your karaf cache! For example, if you change the persisted state directory then we wouldn't write out the initial catalog to it because karaf would have cached those bundles from the previous run.
   label: code-design
2. **body:** Re the last problem, catalog being persisted, I think I'm good with option (1). I was leery of using something that might have collisions (`name_version`, as opposed to the unique ID) but I've grown more comfortable that this isn't an issue in this case. I don't see backwards compatibility being too much of a problem, apart from the obvious. I'm not convinced by your argument against 2. We could say to delete from persistence something if it is a duplicate of a different ID _brooklyn_-managed bundle, then being simply a karaf cache jar wouldn't be enough to block persistence. The slight mess is still that the persisted ID for the default catalog changes on every restart but could live with this if it's easier/safer than (1).
   label: code-design
3. retest this please
4. @ahgittin can you take another look please: gone for option (2) as discussed: only persist the bundle if we actually installed it (so delete the bundle from persisted state if it is a duplicate of another brooklyn-managed-bundle. This was harder than I expected: promotion of an HA hot-standby behaved differently. See unit test and comments for more info.
5. Test failure is real: `org.apache.brooklyn.core.mgmt.ha.HotStandbyTest.testChangeMode`. I've begun to look into that, and will fix it in the morning.
6. @aledsage conflicts with #866 ^ :(#WhyBigPullReqsArentSoBadWhenCodeIsIntertwined (this is causing pain for the reviewer as well as the author)
7. interesting subtlety on ha. nice that black/white list is configurable. minor comment re possibility of unmanaging new bundle rather than old. otherwise good, once conflicts resolved and test fixed.
8. retest this please
9. retest this please failure was the unrelated non-deterministic test `org.apache.brooklyn.core.entity.lifecycle.ServiceStateLogicTest.testStopsNicelyToo`
10. retest this please Excellent, jenkins build successful @ahgittin - I'm getting it to build again to make sure the new tests are reasonably reliable.
11. good stuff, merging

github_pulls_reviews:

1. **body:** This return path only takes effect if passing the `canUpdate()` check on line 379 but presumably behaviour should be the same as for a non-snapshot version? Is the checksum check we do in the other block (line 397 in new money) is equivalent or if we need a super-set check combining that line with `hasBundleOnInstall()`. Probably those checks and the two `ignoringAlreadyInstalled` returns should be moved to before line 379. I'd also rename the new `hasBundleOnInstall` method to `isEquivalentBundleAlreadyOsgiInstalled` or something else a little more descriptive.
   label: code-design
2. I don't like this: it breaks transitivity for the equals method: if `x.equals(y) && `y.equals(z)` then `x.equals(z)` should be true. It fails if `x` and `z` are `BasicManagedBundle` with different checksums or ids, but `y` is a `OsgiBundleWithUrl` with the same name:version and url. However, that was already broken because of the checksum comparison (if an `OsgiBundleWithUrl` impl behaves in the way indicated in the pre-existing comments). Longer term, I lean towards not trying to have equality with `OsgiBundleWithUrl`.
3. agree, should have explicit methods to check or compare the `VersionedName`s where that is the item of interest

jira_issues:

1. **summary:** On restart fails to install catalog bundles due to existing bundles installed with different location
   description: Rebind will always fail on restart due to errors like the following: java.lang.IllegalStateException: Bundle BasicManagedBundle{symbolicName=org.apache.brooklyn.software-cm-ansible, version=1.0.0.SNAPSHOT, url=mvn:org.apache.brooklyn/brooklyn-software-cm-ansible/1.0.0-SNAPSHOT} failed installation: BundleException: Bundle symbolic name and version are not unique: org.apache.brooklyn.software-cm-ansible:1.0.0.SNAPSHOT This seems to have been "caused" by the change: https://github.com/apache/brooklyn-server/pull/862/commits/c4f9d95aa2113a1a5022da17768675599e528dd4 This can be easily tested by starting brooklyn. Stopping brooklyn. And then starting brooklyn.

jira_issues_comments:

1. Full exception: java.lang.IllegalStateException: Bundle BasicManagedBundle{symbolicName=org.apache.brooklyn.software-cm-ansible, version=1.0.0.SNAPSHOT, url=mvn:org.apache.brooklyn/brooklyn-software-cm-ansible/1.0.0-SNAPSHOT} failed installation: BundleException: Bundle symbolic name and version are not unique: org.apache.brooklyn.software-cm-ansible:1.0.0.SNAPSHOT at org.apache.brooklyn.core.mgmt.ha.OsgiArchiveInstaller.install(OsgiArchiveInstaller.java:644) ~ [123:org.apache.brooklyn.core:1.0.0.SNAPSHOT] at org.apache.brooklyn.core.mgmt.ha.OsgiManager.installDeferredStart(OsgiManager.java:359) ~ [123:org.apache.brooklyn.core:1.0.0.SNAPSHOT] at org.apache.brooklyn.core.catalog.internal.CatalogInitialization.installBundle(CatalogInitialization.java:522) ~ [123:org.apache.brooklyn.core:1.0.0.SNAPSHOT] at org.apache.brooklyn.core.catalog.internal.CatalogInitialization.installBundles(CatalogInitialization.java:462) ~ [123:org.apache.brooklyn.core:1.0.0.SNAPSHOT] at org.apache.brooklyn.core.catalog.internal.CatalogInitialization.addPersistedCatalogImpl(CatalogInitialization.java:381) ~ [123:org.apache.brooklyn.core:1.0.0.SNAPSHOT] at org.apache.brooklyn.core.catalog.internal.CatalogInitialization.populateInitialAndPersistedCatalog(CatalogInitialization.java:240) ~[123:org.apache.brooklyn.core:1.0.0.SNAPSHOT] at org.apache.brooklyn.core.mgmt.rebind.RebindIteration.installBundlesAndRebuildCatalog(RebindIteration.java:412) ~ [123:org.apache.brooklyn.core:1.0.0.SNAPSHOT] at org.apache.brooklyn.core.mgmt.rebind.RebindIteration.doRun(RebindIteration.java:254) ~ [123:org.apache.brooklyn.core:1.0.0.SNAPSHOT] at org.apache.brooklyn.core.mgmt.rebind.InitialFullRebindIteration.doRun(InitialFullRebindIteration.java:69) ~ [123:org.apache.brooklyn.core:1.0.0.SNAPSHOT] at org.apache.brooklyn.core.mgmt.rebind.RebindIteration.run(RebindIteration.java:281) ~ [123:org.apache.brooklyn.core:1.0.0.SNAPSHOT] at org.apache.brooklyn.core.mgmt.rebind.RebindManagerImpl.rebindImpl(RebindManagerImpl.java:538) ~ [123:org.apache.brooklyn.core:1.0.0.SNAPSHOT] at org.apache.brooklyn.core.mgmt.rebind.RebindManagerImpl.lambda$rebind$0(RebindManagerImpl.java:494) ~ [123:org.apache.brooklyn.core:1.0.0.SNAPSHOT] at org.apache.brooklyn.util.core.task.BasicExecutionContext$1.call(BasicExecutionContext.java:143) [123:org.apache.brooklyn.core:1.0.0.SNAPSHOT] at org.apache.brooklyn.util.core.task.BasicExecutionContext$1.call(BasicExecutionContext.java:141) [123:org.apache.brooklyn.core:1.0.0.SNAPSHOT] at org.apache.brooklyn.util.core.task.BasicExecutionContext.runInSameThread(BasicExecutionContext.java:227) [123:org.apache.brooklyn.core:1.0.0.SNAPSHOT] at org.apache.brooklyn.util.core.task.BasicExecutionContext.get(BasicExecutionContext.java:141) [123:org.apache.brooklyn.core:1.0.0.SNAPSHOT] at org.apache.brooklyn.core.mgmt.rebind.RebindManagerImpl.rebind(RebindManagerImpl.java:493) [123:org.apache.brooklyn.core:1.0.0.SNAPSHOT] at org.apache.brooklyn.launcher.common.BasicLauncher.startPersistenceWithoutHA(BasicLauncher.java:657) [127:org.apache.brooklyn.launcher-common:1.0.0.SNAPSHOT] at org.apache.brooklyn.launcher.common.BasicLauncher.startPersistence(BasicLauncher.java:620) [127:org.apache.brooklyn.launcher-common:1.0.0.SNAPSHOT] at org.apache.brooklyn.launcher.common.BasicLauncher.handlePersistence(BasicLauncher.java:516) [127:org.apache.brooklyn.launcher-common:1.0.0.SNAPSHOT] at org.apache.brooklyn.launcher.common.BasicLauncher.startPartTwo(BasicLauncher.java:438) [127:org.apache.brooklyn.launcher-common:1.0.0.SNAPSHOT] at org.apache.brooklyn.launcher.osgi.OsgiLauncherImpl.startOsgi(OsgiLauncherImpl.java:116) [274:org.apache.brooklyn.karaf-init:1.0.0.SNAPSHOT] at Proxy3d746dc2_b3b5_47ec_a253_c74925a3baad.startOsgi(Unknown Source) [?:?] at org.apache.brooklyn.launcher.osgi.start.OsgiLauncherCompleter.init(OsgiLauncherCompleter.java:36) [276:org.apache.brooklyn.karaf-start:1.0.0.SNAPSHOT] at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method) ~[?:?] at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62) ~[?:?] at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) ~[?:?] at java.lang.reflect.Method.invoke(Method.java:498) ~[?:?] at org.apache.aries.blueprint.utils.ReflectionUtils.invoke(ReflectionUtils.java:299) [15:org.apache.aries.blueprint.core:1.8.2] at org.apache.aries.blueprint.container.BeanRecipe.invoke(BeanRecipe.java:980) [15:org.apache.aries.blueprint.core:1.8.2] at org.apache.aries.blueprint.container.BeanRecipe.runBeanProcInit(BeanRecipe.java:736) [15:org.apache.aries.blueprint.core:1.8.2] at org.apache.aries.blueprint.container.BeanRecipe.internalCreate2(BeanRecipe.java:848) [15:org.apache.aries.blueprint.core:1.8.2] at org.apache.aries.blueprint.container.BeanRecipe.internalCreate(BeanRecipe.java:811) [15:org.apache.aries.blueprint.core:1.8.2] at org.apache.aries.blueprint.di.AbstractRecipe$1.call(AbstractRecipe.java:79) [15:org.apache.aries.blueprint.core:1.8.2] at java.util.concurrent.FutureTask.run(FutureTask.java:266) [?:?] at org.apache.aries.blueprint.di.AbstractRecipe.create(AbstractRecipe.java:88) [15:org.apache.aries.blueprint.core:1.8.2] at org.apache.aries.blueprint.container.BlueprintRepository.createInstances(BlueprintRepository.java:255) [15:org.apache.aries.blueprint.core:1.8.2] at org.apache.aries.blueprint.container.BlueprintRepository.createAll(BlueprintRepository.java:186) [15:org.apache.aries.blueprint.core:1.8.2] at org.apache.aries.blueprint.container.BlueprintContainerImpl.instantiateEagerComponents(BlueprintContainerImpl.java:704) [15:org.apache.aries.blueprint.core:1.8.2] at org.apache.aries.blueprint.container.BlueprintContainerImpl.doRun(BlueprintContainerImpl.java:410) [15:org.apache.aries.blueprint.core:1.8.2] at org.apache.aries.blueprint.container.BlueprintContainerImpl.run(BlueprintContainerImpl.java:275)

[15:org.apache.aries.blueprint.core:1.8.2] at
org.apache.aries.blueprint.container.BlueprintExtender.createContainer(BlueprintExtender.java:300)
[15:org.apache.aries.blueprint.core:1.8.2] at
org.apache.aries.blueprint.container.BlueprintExtender.createContainer(BlueprintExtender.java:269)
[15:org.apache.aries.blueprint.core:1.8.2] at
org.apache.aries.blueprint.container.BlueprintExtender.createContainer(BlueprintExtender.java:265)
[15:org.apache.aries.blueprint.core:1.8.2] at
org.apache.aries.blueprint.container.BlueprintExtender.modifiedBundle(BlueprintExtender.java:255)
[15:org.apache.aries.blueprint.core:1.8.2] at
org.apache.aries.util.tracker.hook.BundleHookBundleTracker$Tracked.customizerModified(BundleHookBundleTracker.java:500)
[15:org.apache.aries.blueprint.core:1.8.2] at
org.apache.aries.util.tracker.hook.BundleHookBundleTracker$Tracked.customizerModified(BundleHookBundleTracker.java:433)
[15:org.apache.aries.blueprint.core:1.8.2] at
org.apache.aries.util.tracker.hook.BundleHookBundleTracker$AbstractTracked.track(BundleHookBundleTracker.java:725)
[15:org.apache.aries.blueprint.core:1.8.2] at
org.apache.aries.util.tracker.hook.BundleHookBundleTracker$Tracked.bundleChanged(BundleHookBundleTracker.java:463)
[15:org.apache.aries.blueprint.core:1.8.2] at
org.apache.aries.util.tracker.hook.BundleHookBundleTracker$BundleEventHook.event(BundleHookBundleTracker.java:422)
[15:org.apache.aries.blueprint.core:1.8.2] at
org.apache.felix.framework.util.SecureAction.invokeBundleEventHook(SecureAction.java:1179) [?:?] at
org.apache.felix.framework.EventDispatcher.createWhitelistFromHooks(EventDispatcher.java:730) [?:?] at
org.apache.felix.framework.EventDispatcher.fireBundleEvent(EventDispatcher.java:485) [?:?] at
org.apache.felix.framework.Felix.fireBundleEvent(Felix.java:4563) [?:?] at
org.apache.felix.framework.Felix.startBundle(Felix.java:2173) [?:?] at
org.apache.felix.framework.Felix.setActiveStartLevel(Felix.java:1372) [?:?] at
org.apache.felix.framework.FrameworkStartLevelImpl.run(FrameworkStartLevelImpl.java:308) [?:?] at
java.lang.Thread.run(Thread.java:748) [?:?] Caused by: org.osgi.framework.BundleException: Bundle symbolic name and
version are not unique: org.apache.brooklyn.software-cm-ansible:1.0.0.SNAPSHOT at
org.apache.felix.framework.BundleImpl.createRevision(BundleImpl.java:1344) ~[?:?] at
org.apache.felix.framework.BundleImpl.<init>(BundleImpl.java:113) ~[?:?] at
org.apache.felix.framework.Felix.installBundle(Felix.java:3026) ~[?:?] at
org.apache.felix.framework.BundleContextImpl.installBundle(BundleContextImpl.java:167) ~[?:?] at
org.apache.brooklyn.core.mgmt.ha.OsgiArchiveInstaller.install(OsgiArchiveInstaller.java:467) ~[?:?] ... 57 more

2. GitHub user aledsage opened a pull request: https://github.com/apache/brooklyn-server/pull/867 BROOKLYN-546: fix rebind of system bundles You can merge this pull request into a Git repository by running: $ git pull https://github.com/aledsage/brooklyn-server BROOKLYN-546 Alternatively you can review and apply these changes as the patch at: https://github.com/apache/brooklyn-server/pull/867.patch To close this pull request, make a commit to your master/trunk branch with (at least) the following in the commit message: This closes #867 ---- commit e9cdec64572ba1a422dc5d4d67120624f5eac567 Author: Aled Sage <aled.sage@gmail.com> Date: 2017-10-23T12:53:10Z BROOKLYN-546: fix rebind of system bundles ----
3. Github user ahgittin commented on a diff in the pull request: https://github.com/apache/brooklyn-server/pull/867#discussion_r146482165 --- Diff: core/src/main/java/org/apache/brooklyn/core/mgmt/ha/OsgiArchiveInstaller.java --- @@ -379,8 +379,15 @@ private synchronized void close() { if (canUpdate()) { result.bundle = osgiManager.framework.getBundleContext().getBundle(result.getMetadata().getOsgiUniqueUrl()); if (result.getBundle()==null) { - log.warn("Brooklyn thought is was already managing bundle "+result.getMetadata().getVersionedName()+" but it's not installed to framework; reinstalling it"); - updating = false; + if (hasBundleOnInstall(osgiManager, inferredMetadata, zipFile)) { + // e.g. happens if system-bundle is brought under management, and then try to add it again: + // if added from "initial catalog" has it, and then added from persisted state as well. + result.setIgnoringAlreadyInstalled(); + return ReferenceWithError.newInstanceWithoutError(result); --- End diff -- This return path only takes effect if passing the `canUpdate()` check on line 379 but presumably behaviour should be the same as for a non-snapshot version? Is the checksum check we do in the other block (line 397 in new money) is equivalent or if we need a super-set check combining that line with `hasBundleOnInstall()`. Probably those checks and the two `ignoringAlreadyInstalled` returns should be moved to before line 379. I'd also rename the new `hasBundleOnInstall` method to `isEquivalentBundleAlreadyOsgiInstalled` or something else a little more descriptive.
4. Github user aledsage commented on the issue: https://github.com/apache/brooklyn-server/pull/867 @ahgittin Still requires some more testing, but wanted to share this now for feedback. ---- As discussed with @ahgittin, there was an additional rebind issue when loading the initial catalog (and subsequently loading persisted bundles). Karaf will have cached OSGi bundles for that initial catalog. The initial catalog would duplicate some of these - it couldn't tell that the karaf bundle installed on the previous brooklyn run was identical to the bundle it wanted to install (it can't yet see the persisted brooklyn-managed records, and it can't understand osgi location `brooklyn:ID`). Possible solutions for that included: 1) make `brooklyn:ID` osgi location be a checksum (so the same osgi location would be used for two identical brooklyn-managed bundles). 2) clear the karaf cache of brooklyn-managed bundles (e.g. all those with location in the form `brooklyn:ID`), in the knowledge that the initial catalog and persisted state will reinstall them. 3) load initial catalog _after_ persistence (but we expect the initial catalog to tell us whether to remove some items from persistence so this isn't a good option without major re-work). 4) if we're not certain that it's identical, then uninstall and reinstall the bundle (which is what happens currently). This might be a problem if it's a core brooklyn module, but those should have `mvn:` urls so it _can_ confirm whether it's a duplicate). 5) rework rebind / initial catalog load so that we load placeholders for bundles that have been persisted (enough to do the deduping or install things referenced in initial catalog load). We opted for doing (1) and (4). --- Another issue (FAO @ahgittin) is that every time Brooklyn was started, we'd add to the persisted state another copy of each core brooklyn bundle! This is because the initial catalog references things like `org.apache.brooklyn.policy`. The `ManagedBundle` we create to represent it gets an auto-generated unique id. We see that the

bundle is already installed, so we don't re-install it in karaf (good). But we do still persist it to the `/bundles` persisted state sub-directory. I've worked around that by black-listing `org.apache.brooklyn.*` from being persisted (see `OsgiArchiveInstaller.isBlacklistedForPersistence(ManagedBundle)`. The assumption is that those will always ship with Brooklyn - we should use just the version that is shipped with the given release of Brooklyn. --- Another issue (similar, not yet fixed) is that we persist a new identical copy of `brooklyn-default-karaf-catalog` every time we restart Brooklyn. This is because the `ManagedBundle` for the initial catalog has an auto-generated unique id. As above, we do not install the duplicate bundle, but we still persist it. Luckily rebind is not as bad as you might think! It spots that an identical bundle is already installed, so doesn't install it or parse the catalog.bom of every copy that is in persisted state. Possible solutions for this include: 1) changing the persistence file/object names to use the `symbolicName_version.jar` format (similar naming to the old `/catalog` directory). 2) only persist the bundle if we actually installed it (i.e. it wasn't a duplicate). 3) search persisted state for a duplicate, and don't add if it already exists. Option (1) is very tempting, given that is it's logical unique identity: we shouldn't be persisting (and thus installing) multiple things with the same name:version. We'd need to be careful with backwards compatibility if someone has `/bundles` that uses the unique ids. We could later add additional checks to only replace the persisted file if its checksum had actually changed. Option (2) feels wrong - what is persisted depends on what is in your karaf cache! For example, if you change the persisted state directory then we wouldn't write out the initial catalog to it because karaf would have cached those bundles from the previous run.

5. Github user ahgittin commented on the issue: https://github.com/apache/brooklyn-server/pull/867 Re the last problem, catalog being persisted, I think I'm good with option (1). I was leery of using something that might have collisions (`name_version`, as opposed to the unique ID) but I've grown more comfortable that this isn't an issue in this case. I don't see backwards compatibility being too much of a problem, apart from the obvious. I'm not convinced by your argument against 2. We could say to delete from persistence something if it is a duplicate of a different ID _brooklyn_-managed bundle, then being simply a karaf cache jar wouldn't be enough to block persistence. The slight mess is still that the persisted ID for the default catalog changes on every restart but could live with this if it's easier/safer than (1).
6. Github user aledsage commented on the issue: https://github.com/apache/brooklyn-server/pull/867 retest this please
7. Github user aledsage commented on a diff in the pull request: https://github.com/apache/brooklyn-server/pull/867#discussion_r147279623 --- Diff: core/src/main/java/org/apache/brooklyn/core/typereg/BasicManagedBundle.java --- @@ -135,6 +145,9 @@ public boolean equals(Object obj) { // this makes equality with other OsgiBundleWithUrl items symmetric, // but for two MB's we look additionally at checksum if (!Objects.equal(checksum, ((ManagedBundle)other).getChecksum())) return false; + + // only equal if have the same ManagedBundle uid; important for persistence.changeListener().unmanage() + if (!Objects.equal(getId(), ((ManagedBundle)other).getId())) return false; --- End diff -- I don't like this: it breaks transitivity for the equals method: if `x.equals(y) && `y.equals(z)` then `x.equals(z)` should be true. It fails if `x` and `z` are `BasicManagedBundle` with different checksums or ids, but `y` is a `OsgiBundleWithUrl` with the same name:version and url. However, that was already broken because of the checksum comparison (if an `OsgiBundleWithUrl` impl behaves in the way indicated in the pre-existing comments). Longer term, I lean towards not trying to have equality with `OsgiBundleWithUrl`.
8. Github user aledsage commented on the issue: https://github.com/apache/brooklyn-server/pull/867 @ahgittin can you take another look please: gone for option (2) as discussed: only persist the bundle if we actually installed it (so delete the bundle from persisted state if it is a duplicate of another brooklyn-managed-bundle. This was harder than I expected: promotion of an HA hot-standby behaved differently. See unit test and comments for more info.
9. Github user aledsage commented on the issue: https://github.com/apache/brooklyn-server/pull/867 Test failure is real: `org.apache.brooklyn.core.mgmt.ha.HotStandbyTest.testChangeMode`. I've begun to look into that, and will fix it in the morning.
10. Github user ahgittin commented on a diff in the pull request: https://github.com/apache/brooklyn-server/pull/867#discussion_r147382359 --- Diff: core/src/main/java/org/apache/brooklyn/core/typereg/BasicManagedBundle.java --- @@ -135,6 +145,9 @@ public boolean equals(Object obj) { // this makes equality with other OsgiBundleWithUrl items symmetric, // but for two MB's we look additionally at checksum if (!Objects.equal(checksum, ((ManagedBundle)other).getChecksum())) return false; + + // only equal if have the same ManagedBundle uid; important for persistence.changeListener().unmanage() + if (!Objects.equal(getId(), ((ManagedBundle)other).getId())) return false; --- End diff -- agree, should have explicit methods to check or compare the `VersionedName`s where that is the item of interest
11. Github user ahgittin commented on the issue: https://github.com/apache/brooklyn-server/pull/867 @aledsage conflicts with #866 ^ :(#WhyBigPullReqsArentSoBadWhenCodeIsIntertwined (this is causing pain for the reviewer as well as the author)
12. Github user ahgittin commented on the issue: https://github.com/apache/brooklyn-server/pull/867 interesting subtlety on ha. nice that black/white list is configurable. minor comment re possibility of unmanaging new bundle rather than old. otherwise good, once conflicts resolved and test fixed.
13. Github user ahgittin commented on the issue: https://github.com/apache/brooklyn-server/pull/867 retest this please
14. Github user aledsage commented on the issue: https://github.com/apache/brooklyn-server/pull/867 retest this please failure was the unrelated non-deterministic test `org.apache.brooklyn.core.entity.lifecycle.ServiceStateLogicTest.testStopsNicelyToo`
15. Github user aledsage commented on the issue: https://github.com/apache/brooklyn-server/pull/867 retest this please Excellent, jenkins build successful @ahgittin - I'm getting it to build again to make sure the new tests are reasonably reliable.
16. Github user ahgittin commented on the issue: https://github.com/apache/brooklyn-server/pull/867 good stuff, merging
17. Github user asfgit closed the pull request at: https://github.com/apache/brooklyn-server/pull/867