Building and Installing CORD Software

Zack Williams, zdw@opennetworking.org

CORD Build Nov. 7-9, 2017

An Operator Led Consortium

What is a build system?

Build system is **code** for the **process** that assembles a project

Building a CORD POD is a complex integration process

Build system encapsulates and automates all this complexity

Overview

- Where we were in the past
- Goals of the 4.0 build system
- Design of the 4.0 build system
- Build Tools
- Demo
- Future Plans

Where we were before CORD 4.0

Earlier versions of CORD had two (or more) different entry points:

- Ansible-based for developer workflows
- Gradle-based for deployment workflows

Configuration spread across system, hard to change

Docker images could be different than code checkout

Full POD builds were R-CORD centric, hard to extend, hard to resume a build

Goals for the 4.0 build system

- Correctness
- Ease of use
- Incremental build process
- Extensibility/Modularity
- Documentation
- Speed

Design of the 4.0 build system

New Concepts in 4.0:

- Separation of build into mix-and-match **Scenario** and **Profile**
- **POD config** for per-POD configuration
- Workflow is the same for both development and deployment
- Incremental build that can be resumed after error

Kept most of the underlying build code

POD Config

POD Config: a single configuration file for the entire build system.

YAML file that is imported by all runs of Ansible and Vagrant, and used to configure Makefile.

Stored in `build/podconfig/<profile>_<scenario>.yml`

At a minimum, specifies the Scenario and Profile:

```
# rcord-mock Pod Config
# Creates a single-node mock R-CORD pod
cord_scenario: mock
cord_profile: rcord
```

Scenarios

Scenarios: The hardware topology and software environment to deploy.

Determines the “feature matrix” installed *external* to XOS, the inventory of nodes, and how those nodes are used

	XOS	ONOS	MaaS	OpenStack	Build VM
mock	✔				
single	✔	✔			
cord	✔	✔	✔	✔	✔
preppedpod	✔	✔		✔	

Scenarios, cont.

Scenarios are stored in `build/scenarios/<name>/`

- `config.yml` - Configuration
- `Vagrantfile` - VM definitions

Profiles

Profiles: The service graph that makes up a CORD service set

The traditional R-CORD, E-CORD, M-CORD, etc. sets of services and how they're connected together.

Stored in:
`build/platform-install/profile_manifests/<name>.yml`

Build Contexts

The build has multiple command execution contexts:

- config - where configuration is generated
- build - where containers and other software is built
- head - the CORD head node
- compute - OpenStack compute nodes

These all correspond to the Ansible inventory groups.

A node may belong to multiple contexts.

Build Configuration

CORD is highly configurable - >150 different settings to control the build of CORD. We autogenerate this documentation:

https://guide.opencord.org/build_glossary.html

Same variables used across POD Config, Scenario, or Profile

Build Config Hierarchy

	var1	var2	var3	var4	var5
POD Config	podconfig		podconfig		
Scenario		scenario			
Profile	profile	profile		profile	
Default	default	default	default	default	default
Result	**podconfig**	**scenario**	**podconfig**	**profile**	**default**

Override order is POD Config > Scenario > Profile > Default

Build Tools

Top-level tool is `make`

Individual make targets invokes other tools:

- Vagrant
- Ansible
- ImageBuilder
- CoreBuilder

Make

Targets are used for each task, and create **Milestone** files which are indicate task completion. Targets invoked by dependencies.

~ 70 targets for CORD (build, development, utility)

Dynamic target dependencies set with the `*_prereqs` variables, which are added to `build/genconfig/config.mk`

Ansible

Primary tool for installing and configuring CORD components.

Modular sets of tasks grouped into roles, which are grouped into playbooks.

Heavily used across the build system, and within XOS synchronizers

- `make config` invokes Ansible to generate the files in `build/genconfig/`

Primary set of playbooks is in `build/platform-install`

Build Commands

Every build type is just two `make` targets:

```
make PODCONFIG=<podconfig file> config
make build
```

Target: config

```
make PODCONFIG=<podconfig file> config
```

`config` target generates files in `build/genconfig`:

- `config.yml` - Merge of POD config and Scenario
- `config.mk` - Makefile dependencies and variables
- `inventory.ini` - Ansible inventory of nodes per build context
- `cord_profile`, `cord_scenario` - selected profile/scenario

Target: build

`make build`

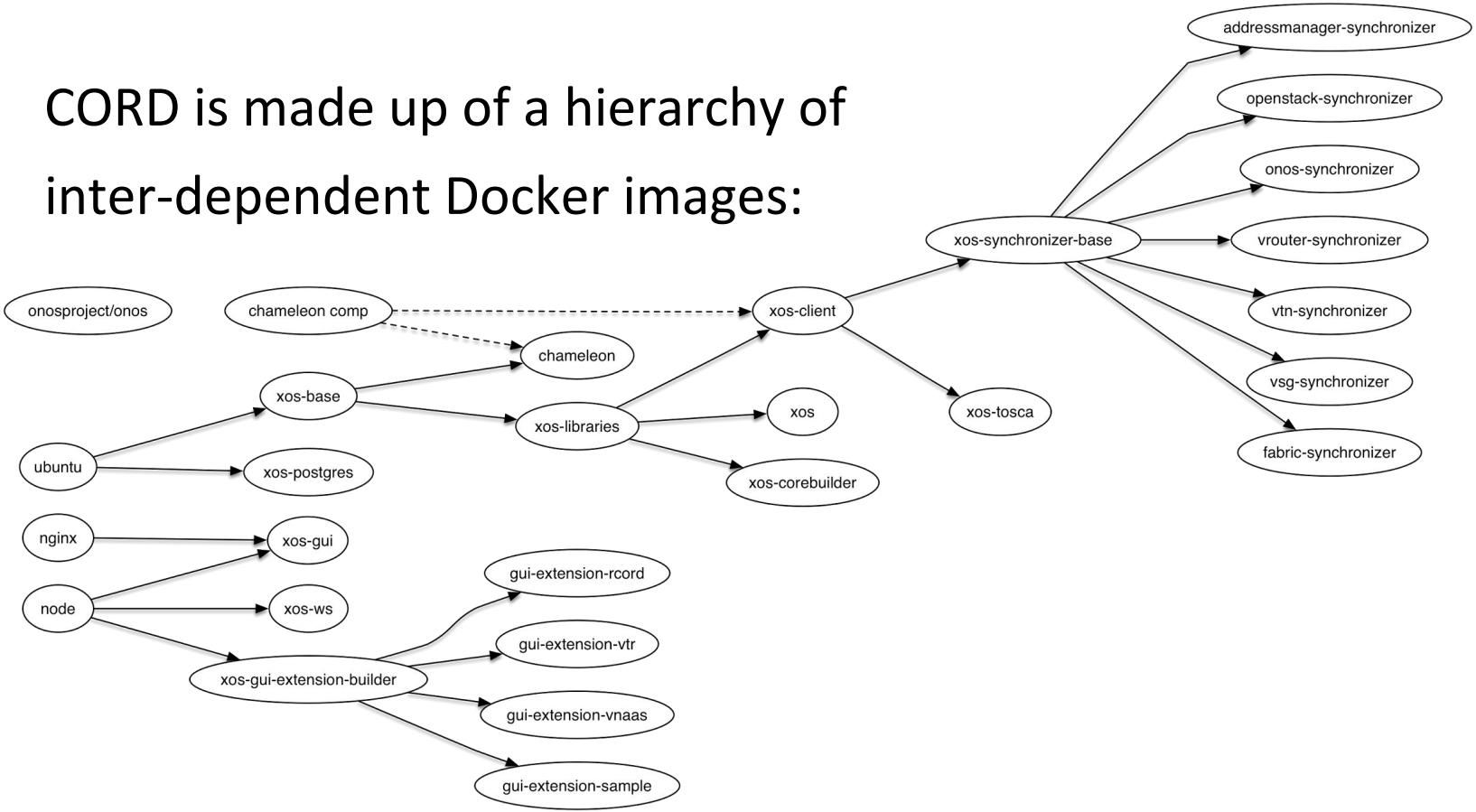
`build_targets` variable from the Scenario determines the final target of the build. Other targets invoked based on dependencies.

Target output is logged in `build/logs` , named with a timestamp and the name of the target.

Successful target runs leave milestones behind in build/milestones

ImageBuilder

CORD is made up of a hierarchy of inter-dependent Docker images:



ImageBuilder, cont.

Custom tool that builds Dockerfiles into docker images that is `git`, `repo`, and dependency aware, and tags/labels appropriately.

Downloads pre-built images from Docker Hub if source tree is unmodified. Tags/labels based on git commit hash.

If an image needs to be built, build logs are stored in `build/image_logs` on the build node

CoreBuilder

Takes a set of onboarding recipes and generates the data model based on `xproto` specifications

Static build-time tool, run by an Ansible playbook that creates the `xos-core` Docker image

Expected to be replaced by dynamic onboarding "soon"

Development Targets

For developing CORD, there are targets that help developers iterate.

Example workflow:

- `make build`
- `<change source code>`
- `make xos-teardown`
- `make build`

See Documentation for other development loops.

Demo

Performance Improvements

Build time is down by ~46%, for the full POD end-to-end testing case:

3.0 CiaB (gradle): 159m42.723s

4.0 CiaB (make): 86m17.056s

4.1 and 5.0 will be even faster

Future Plans

Adding new modular features for Scenarios to include

- Kubernetes
- Docker Swarm

Turning build system components into XOS services/synchronizers

Testing for target dependencies as Scenarios increase in number

How to contribute

Build CORD and give feedback on the process

Documentation review and fixes

Adding new Scenario and Profiles

New Platform/Architecture support

Add or Extend new features for Scenarios

Build to> XOS service migration

Questions?

Thanks!