µONOS for Developers

Andrea Campanella, Jordan Halterman
Open Networking Foundation
<andrea,jordan>@opennetworking.org
Overview

• μONOS architecture overview
• Code structure
• Development environment
• Deployment
• Development workflow
• How to contribute
What is µONOS?

- µONOS is the next generation architecture of ONOS
- Aims to provide a comprehensive platform for operations
  - configuration, control, monitoring, verification, live update, diagnostics
- Aims to provide first-class support for 5G RAN edge
- Based on µ-services, gRPC interfaces, next-gen SDN interfaces
  - e.g. gNMI, gNOI, P4Runtime, gRIBI, etc.
- Cloud-native (Kubernetes) and aimed at edge-cloud
µONOS GitHub Repositories

- Multiple repos that reflect the component architecture
- Components built/packaged independently
- Current repos include:
  - `onos-config`, `onos-topo`, `onos-control`,
  - `onos-gui`, `onos-cli`, `onos-test`
- More refactoring to follow
- Everything hosted under https://github.com/onosproject
uONOS Code Architecture

**onos-gui**
Graphical user interface for ONOS (μONOS Architecture)
- Javascript
- Apache-2.0
- Updated 11 hours ago

**onos-ztp**
Zero-Touch-Provisioning subsystem for ONOS (μONOS Architecture)
- Go
- Apache-2.0
- Updated yesterday

**onos-test**
Integration test infrastructure for ONOS (μONOS Architecture)
- Go
- Apache-2.0
- Updated yesterday

**onos-topo**
Topology subsystem for ONOS (μONOS Architecture)
- Go
- Apache-2.0
- Updated 3 days ago

**onos-config**
Configuration subsystem for ONOS (μONOS Architecture)
- Go
- Apache-2.0
- Updated on Jul 11

**onos-cli**
Command-line interface for ONOS (μONOS Architecture)
- Go
- Apache-2.0
- Updated 10 days ago

**onos-control**
Control subsystem for ONOS (μONOS Architecture)
- Go
- Apache-2.0
- Updated 12 days ago

**simulators**
Code for simulating various device and orchestration entities with which ONOS interacts, e.g. gNMI, gNOI, P4Runtime
- Go
- Apache-2.0
- Updated 15 days ago

**app-registry**
Hosts project for a portal that tracks ONOS and μONOS applications.
- TypeScript
- Apache-2.0
- Updated on Jul 23

**onos-ran**
RAN subsystem for ONOS (μONOS Architecture)
- Apache-2.0
- Updated on Jul 11

https://github.com/onosproject
Consistency

Tools, operations, and processes are consistent across all subsystems and their repositories

Learn the workflow and tools for one service: contribute to all the microservices/repos
μONOS Deployment

- gRPC
- app X
- gRPC
- app Y
- gRPC
- gRPC
- ztp
- gNMI/gNOI/P4Runtime
- gui
- gRPC
- cli
- gRPC
- NetModel
topology
- NetCertscerts
- NetDisco
discovery
- k/v
- gRPC
- NetCertscerts
- gNMI/gNOI
- P4Runtime
- gNMI/gNOI
- P4Runtime
- P4Runtime
- P4Runtime
- gRNI
gRNI
- gRNI
- ran
- SD-RAN
- adapter
- various protocols
- discovery
- network
- k8s
μONOS Deployment

Workload Status

- Deployments: 100.0%
- Pods: 100.0%
- Replica Sets: 100.0%
- Stateful Sets: 100.0%

Deployments

<table>
<thead>
<tr>
<th>Name</th>
<th>Labels</th>
<th>Pods</th>
<th>Age</th>
<th>Images</th>
</tr>
</thead>
<tbody>
<tr>
<td>onos-ztp</td>
<td>app: onos</td>
<td>resource: onos-ztp</td>
<td>1 / 1</td>
<td>3 hours</td>
</tr>
<tr>
<td>onos-cli</td>
<td>-</td>
<td>1 / 1</td>
<td>3 hours</td>
<td>onosproject/onos-cli/latest</td>
</tr>
<tr>
<td>onos-gui</td>
<td>-</td>
<td>1 / 1</td>
<td>3 hours</td>
<td>onosproject/onos-gui/latest</td>
</tr>
<tr>
<td>onos-config</td>
<td>-</td>
<td>1 / 1</td>
<td>3 hours</td>
<td>onosproject/onos-config/latest</td>
</tr>
<tr>
<td>onos-config-envoy</td>
<td>-</td>
<td>1 / 1</td>
<td>3 hours</td>
<td>envoyproxy/envoy-alpine.latest</td>
</tr>
<tr>
<td>onos-topo-envoy</td>
<td>-</td>
<td>1 / 1</td>
<td>3 hours</td>
<td>envoyproxy/envoy-alpine.latest</td>
</tr>
<tr>
<td>onos-topo</td>
<td>-</td>
<td>2 / 2</td>
<td>3 hours</td>
<td>onosproject/onos-topo/latest</td>
</tr>
<tr>
<td>atomix-controller</td>
<td>-</td>
<td>1 / 1</td>
<td>3 hours</td>
<td>atomix/atomix-k8s-controller.latest</td>
</tr>
</tbody>
</table>
# µONOS Deployment

<table>
<thead>
<tr>
<th>Name</th>
<th>Labels</th>
<th>Cluster IP</th>
<th>Internal Endpoints</th>
<th>External Endpoints</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>onos-ztp</td>
<td>-</td>
<td>10.110.156.202</td>
<td>onos-ztp, onos-demo:5150 TCP, onos-ztp, onos-demo:0 TCP</td>
<td>-</td>
<td>3 hours</td>
</tr>
<tr>
<td>onos-gui</td>
<td>-</td>
<td>10.100.79.38</td>
<td>onos-gui, onos-demo:80 TCP, onos-gui, onos-demo:0 TCP</td>
<td>-</td>
<td>3 hours</td>
</tr>
<tr>
<td>onos-config</td>
<td>-</td>
<td>10.107.196.83</td>
<td>onos-config, onos-demo:5150 TCP, onos-config, onos-demo:0 TCP</td>
<td>-</td>
<td>3 hours</td>
</tr>
<tr>
<td>onos-config-envoy</td>
<td>-</td>
<td>10.100.63.83</td>
<td>onos-config-envoy, onos-demo:8080 TCP, onos-config-envoy, onos-demo:0 TCP</td>
<td>-</td>
<td>3 hours</td>
</tr>
<tr>
<td>onos-topo</td>
<td>-</td>
<td>10.107.254.44</td>
<td>onos-topo, onos-demo:5150 TCP, onos-topo, onos-demo:0 TCP</td>
<td>-</td>
<td>3 hours</td>
</tr>
<tr>
<td>onos-topo-envoy</td>
<td>-</td>
<td>10.105.237.232</td>
<td>onos-topo-envoy, onos-demo:8080 TCP, onos-topo-envoy, onos-demo:0 TCP</td>
<td>-</td>
<td>3 hours</td>
</tr>
<tr>
<td>raft</td>
<td>-</td>
<td>10.102.199.255</td>
<td>raft, onos-demo:5678 TCP, raft, onos-demo:0 TCP</td>
<td>-</td>
<td>3 hours</td>
</tr>
<tr>
<td>raft-1</td>
<td>app: atomix, group: raft</td>
<td>10.97.128.253</td>
<td>raft-1, onos-demo:5678 TCP, raft-1, onos-demo:0 TCP</td>
<td>-</td>
<td>3 hours</td>
</tr>
</tbody>
</table>
μONOS and Atomix

- Cloud native database
  - Kubernetes controller for database management
  - Atomix nodes for persistence and replication
- Implements Protobuf data structures API
- Atomix Go client used in Go services for persistence/fault tolerance
- https://github.com/atomix
Golang Language

- Significant momentum in ecosystem
- Excellent integration with gRPC and native Supports streaming APIs
- no JVM or JIT compiler
- Go has garbage collection but is less prone to memory leaks, faster and safer code development
- GO GC cycles do require STW pauses which does have some impact with respect to apps that have real-time requirements; however:
  - Current (2017+) releases of Go runtime have ~500μs STW pauses
- Real-time sensitive portions in C/C++ if required and Go runtime using foreign function interface (FFI)
Developer Environment

Standard Go project structure
- [https://github.com/golang-standards/project-layout](https://github.com/golang-standards/project-layout)

Dependency management and build done through go modules
- GO111MODULE=on
- use go.sum and go.mod
Developer Environment (Docker)

Go code runs in a docker container

Tagged images easily downloadable from dockerhub:
https://hub.docker.com/u/onosproject

Build your own images as you change code: `make images` in all of the uploaded repos
Developer Environment

Kubernetes manages the deployment of docker containers:
- Helm Charts
- Onit test tools

Kubernetes is usually installed/run:
- bare-metal
- Kind on any docker capable env: https://kind.sigs.k8s.io
Developer Environment

In short:
ONIT Overview

ONIT: ONOS Integration Test framework

• CLI for development and testing on Kubernetes
• Deploy µONOS in a Kubernetes cluster
• Run integration tests:
  • end to end “black-box” testing of µONOS subsystems
  • E.g. subscription, get, set, model plugins …
• Run benchmarks
• Remote debugging via delve
• Deploy and manage applications inside k8s

https://github.com/onosproject/onos-test
ONIT Development Workflow
ONIT Development Workflow

Usage:
onit [command]

Available Commands:
add
completion
create
debug
delete
defetch
getch
get
help
help-cli
remove
run
set
ssh

Flags:
-h, --help help for onit

Use "onit [command] --help" for more information about a command.

> onos-test git:(master) kubectl get pods -n onos-test

NAME                    READY STATUS    RESTARTS AGE
atomix-controller-4b6d87f54f-72c92 1/1 Running   0 52s
atomix-cli-6899b56c7-4pprk       1/1 Running   0 13s
atomix-config-77dbf5b06f-6f5sv   1/1 Running   0 28s
atomix-config-88b8f7d6-smdgb     1/1 Running   0 27s
atomix-gui-679b87c7c-2zggs       1/1 Running   0 14s
atomix-topo-7f0d4484e8-6534fgr   1/1 Running   0 36s
atomix-topo-8899b5d8-66l1v       1/1 Running   0 36s
raft-1-0                     1/1 Running   0 45s

> onos-test git:(master) onit -h
ONIT Development Workflow
ONIT Development Workflow

```
* onos-test git:(master) onit onos-cli
  $ onos topo get device
Get topology resources

Usage:
  onos topo get [command]

Available Commands:
  device get a device

Flags:
  -h, --help help for get

Use "onos topo get [command] --help" for more information about a command.
  $ onos topo get devices
  $ ADDRESS VERSION
  $ onos topo add device test --address test:1234 --version
```
ONIT Development Workflow
ONIT Development Workflow
ONIT Development Workflow
ONIT Development Workflow

```bash
# onos-topo git:(master) git checkout -b device-type-filter
Switched to a new branch 'device-type-filter'

# onos-topo git:(device-type-filter) git status
On branch device-type-filter
Changes not staged for commit:
  (use 'git add <file>...' to update what will be committed)
  (use 'git checkout -- <file>...' to discard changes in working directory)

    modified:   pkg/cli/device.go

no changes added to commit (use 'git add' and/or 'git commit -a')

# onos-topo git:(device-type-filter) git commit -am "Add device type filter to devices command."

1 file changed, 8 insertions(+), 4 deletions(-)

# onos-topo git:(device-type-filter) git push origin device-type-filter
Enumerating objects: 9, done.
Counting objects: 100K (6/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (5/5), 563 bytes | 563.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0)
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
remote: remote: Create a pull request for 'device-type-filter' on GitHub by visiting:
remote: https://github.com/kwujo/onos-topo/pull/new/device-type-filter
remote: To github.com:kwujo/onos-topo.git
  * [new branch] device-type-filter -> device-type-filter
# onos-topo git:(device-type-filter)
```
ONIT Development Workflow

Open a pull request
Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

Able to merge. These branches can be automatically merged.

Add device type filter to devices command

Commit on Sep 10, 2019

Add device type filter to devices command.
Contributing to µONOS

Follow the developer workflow:
https://github.com/onosproject/onos-config/blob/master/docs/dev_workflow.md

4 main sections

• Fork the project, download the code, create a local development environment
• Code, Code, Code then build and test
• Submit a pull request
• work with the community on comments and enhancements
• Get your PR merged and see your code in action
µONOS uses **GitHub as an all in one integrated tool** for code, issues, comments, documentation, PR, CI/CD:

- Avid tool multiplication and learning curve
- exploit all in one integration
- one stop shop for everything related to the project
- automated CI/CD integration on PRs with Travis
Github and Travis

Travis, simple, well known, ubiquitous CI/CD tool

μONOS Travis workflow:
- Build docker images based on submitted changes
- Deploy Kind cluster with new docker images
- Integrations tests
- License
- Code checkstyle and lint (golangci-lint)
How to get involved

• Join #micro-onos channel on onosproject.slack.com
• Attend ONOS TST meetings
  • bi-weekly Wednesdays at 9:00 PST/PDT
• Fork and send pull-requests to https://github.com/onosproject repositories
• Participate in onos-dev@onosproject.org mailing list
Thank You

Follow Up Links:

µONOS repositories
Atomix repositories