



# μONOS for Developers

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

# Overview

- $\mu$ ONOS architecture overview
- Code structure
- Development environment
- Deployment
- Development workflow
- How to contribute

# What is $\mu$ ONOS?

- $\mu$ ONOS 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  $\mu$ -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 ( $\mu$ ONOS Architecture)

JavaScript Apache-2.0 3 2 2 0 Updated 11 ho

## onos-ztp

Zero-Touch-Provisioning subsystem for ONOS ( $\mu$ ONOS Architecture)

Go Apache-2.0 8 0 0 0 Updated yesterday

## onos-test

Integration test infrastructure for ONOS ( $\mu$ ONOS Architecture)

Go Apache-2.0 7 4 4 2 Updated yesterday

## onos-topo

Topology subsystem for ONOS ( $\mu$ ONOS Architecture)

Go Apache-2.0 8 3 1 0 Updated 3 days ago

## onos-config

Configuration subsystem for ONOS ( $\mu$ ONOS Architecture)

Go Apache-2.0 16 16 19 (4 issues need help) 1

## onos-cli

Command-line interface for ONOS ( $\mu$ ONOS Architecture)

Go Apache-2.0 6 1 0 1 Updated 10 days ago

## onos-control

Control subsystem for ONOS ( $\mu$ ONOS Architecture)

Go Apache-2.0 5 5 0 1 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 5 2 2 (1 issue needs help) 1 Updated 15 c

## app-registry

Hosts project for a portal that tracks ONOS and  $\mu$ ONOS applications.

TypeScript Apache-2.0 3 1 0 0 Updated on Jul 23

## onos-ran

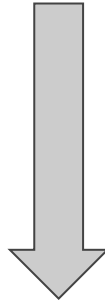
RAN subsystem for ONOS ( $\mu$ ONOS Architecture)

Apache-2.0 1 0 0 0 Updated on Jul 11

<https://github.com/onosproject>

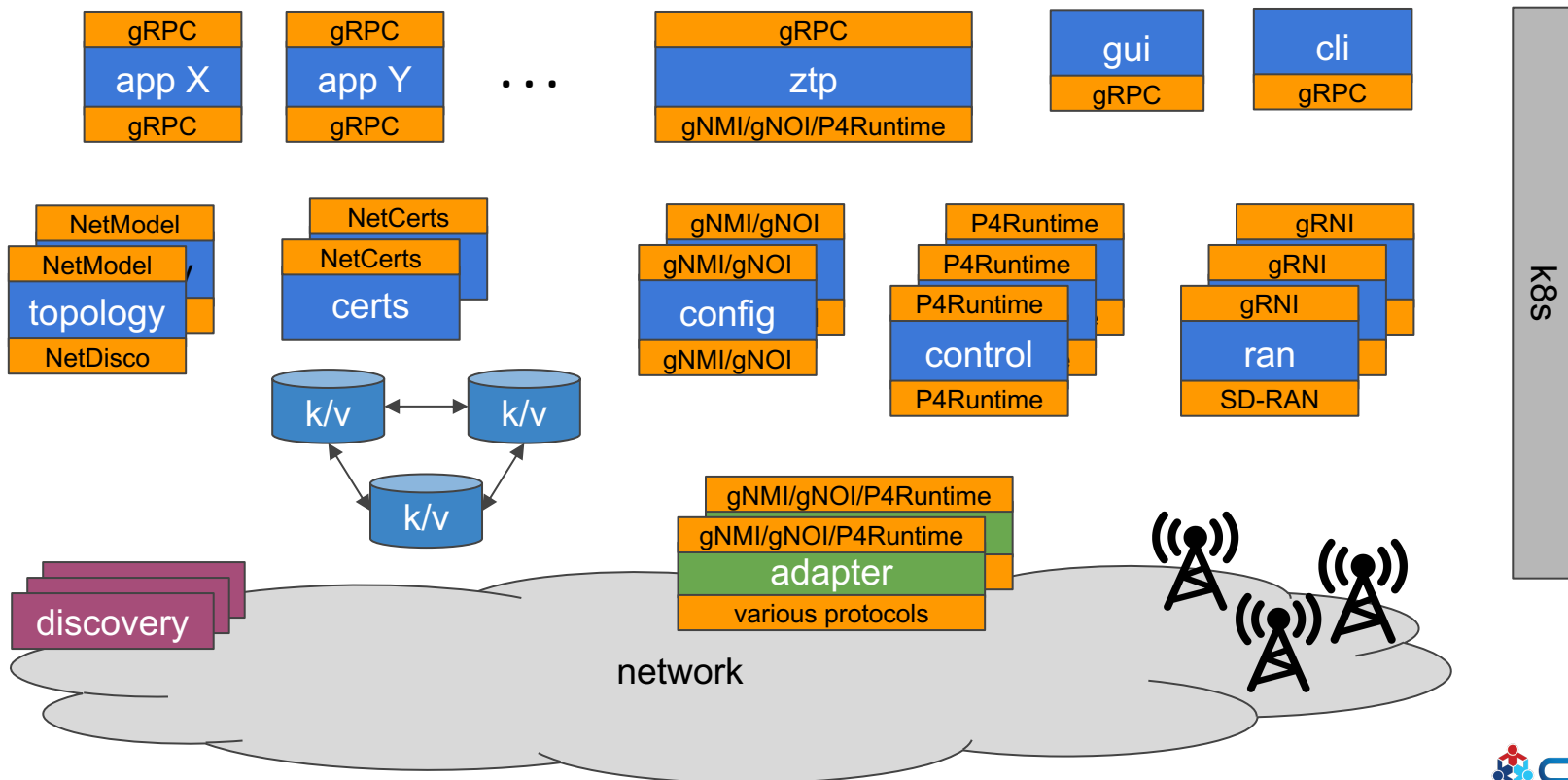
# Consistency

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



# μONOS Deployment

## Workload Status



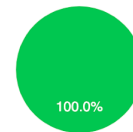
Deployments



Pods



Replica Sets



Stateful Sets

## Deployments



















Name	Labels	Pods	Age ↑	Images
onos-ztp	app: onos resource: onos-ztp <a href="#">Show all</a>	1 / 1	<a href="#">3 hours</a>	onosproject/onos-ztp:latest
onos-cli	-	1 / 1	<a href="#">3 hours</a>	onosproject/onos-cli:latest
onos-gui	-	1 / 1	<a href="#">3 hours</a>	onosproject/onos-gui:latest
onos-config	-	1 / 1	<a href="#">3 hours</a>	onosproject/onos-config:latest
onos-config-envoy	-	1 / 1	<a href="#">3 hours</a>	envoyproxy/envoy-alpine:latest
onos-topo-envoy	-	1 / 1	<a href="#">3 hours</a>	envoyproxy/envoy-alpine:latest
onos-topo	-	2 / 2	<a href="#">3 hours</a>	onosproject/onos-topo:latest
atomix-controller	-	1 / 1	<a href="#">3 hours</a>	atomix/atomix-k8s-controller:latest

1 - 8 of 8





# μONOS Deployment

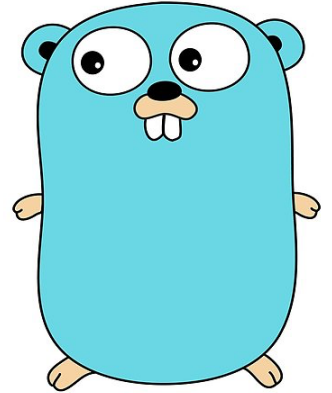
Services							
Name	Labels	Cluster IP	Internal Endpoints	External Endpoints	Age ↑		
 <a href="#">onos-ztp</a>	-	10.110.156.202	onos-ztp.onos-demo:5150 TCP onos-ztp.onos-demo:0 TCP	-	<a href="#">3 hours</a>		
 <a href="#">onos-gui</a>	-	10.100.79.38	onos-gui.onos-demo:80 TCP onos-gui.onos-demo:0 TCP	-	<a href="#">3 hours</a>		
 <a href="#">onos-config</a>	-	10.107.196.83	onos-config.onos-demo:5150 TCP onos-config.onos-demo:0 TCP	-	<a href="#">3 hours</a>		
 <a href="#">onos-config-envoy</a>	-	10.100.63.83	onos-config-envoy.onos-demo:8080 TCP onos-config-envoy.onos-demo:0 TCP	-	<a href="#">3 hours</a>		
 <a href="#">onos-topo</a>	-	10.107.254.44	onos-topo.onos-demo:5150 TCP onos-topo.onos-demo:0 TCP	-	<a href="#">3 hours</a>		
 <a href="#">onos-topo-envoy</a>	-	10.105.237.232	onos-topo-envoy.onos-demo:8080 TCP onos-topo-envoy.onos-demo:0 TCP	-	<a href="#">3 hours</a>		
 <a href="#">raft</a>	-	10.102.199.255	raft.onos-demo:5678 TCP raft.onos-demo:0 TCP	-	<a href="#">3 hours</a>		
 <a href="#">raft-1</a>	<span>app: atomix</span> <span>group: raft</span> <span>Show all</span>	10.97.128.253	raft-1.onos-demo:5678 TCP raft-1.onos-demo:0 TCP	-	<a href="#">3 hours</a>		

# 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)



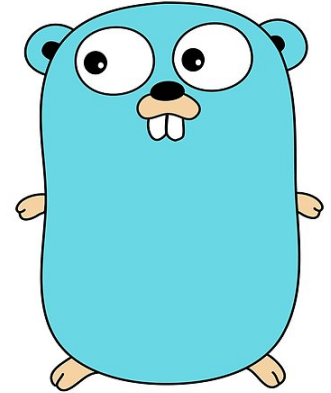
# Developer Environment

Standard Go project structure

- <https://github.com/golang-standards/project-layout>

Dependency management and build done through go modules

- `GO111MODULE=on`
- use `go.sum` and `go.mod`



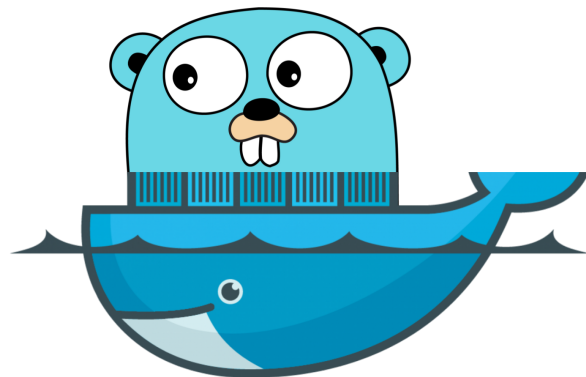
# Developer Environment (Docker)

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



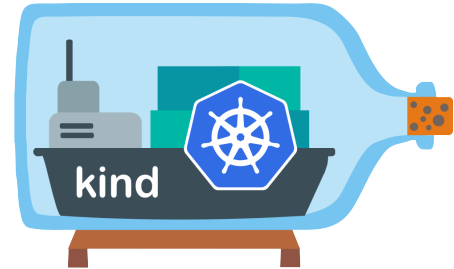
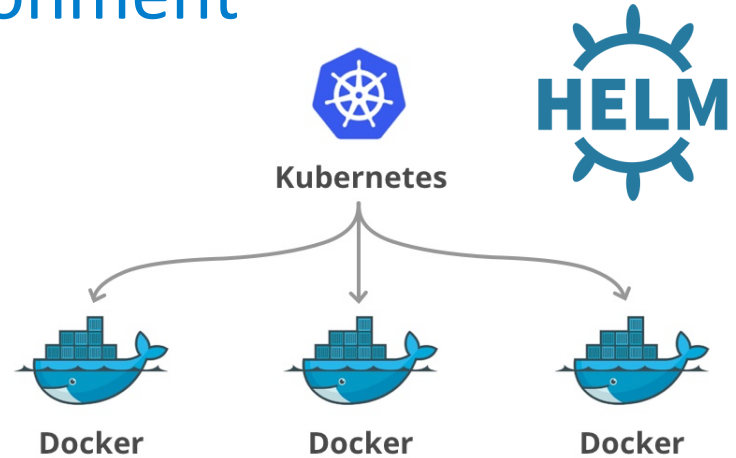
# Developer Environment

Kubernetes manages the deployment of docker containers:

- Helm Charts
- Onit test tools

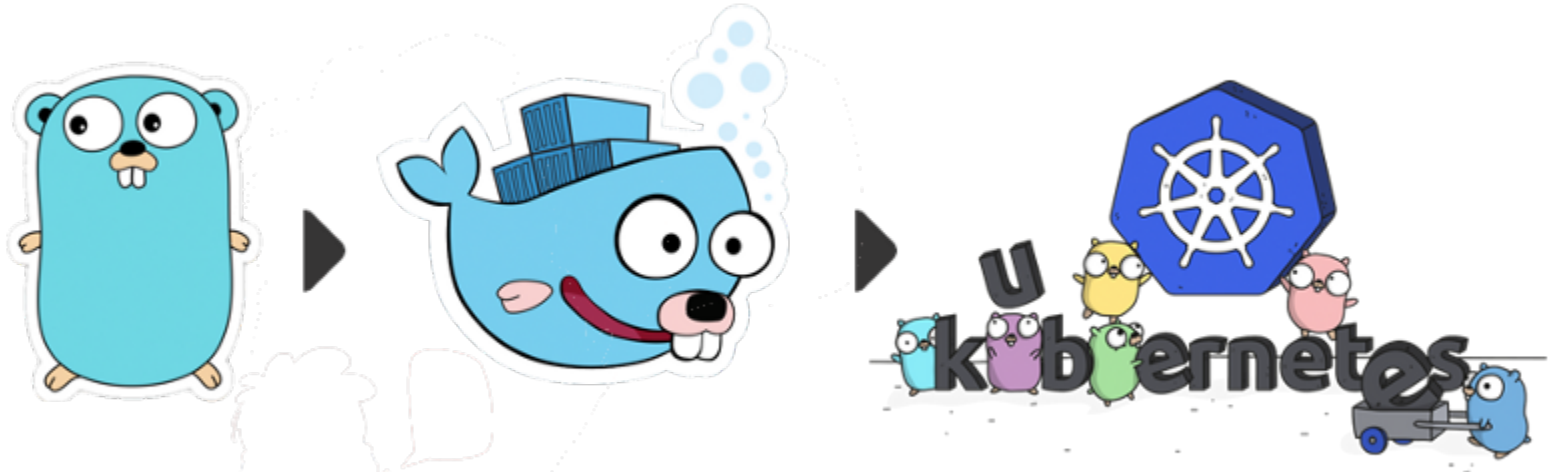
Kubernetes is usually installed/run:

- 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  $\mu$ ONOS in a Kubernetes cluster
- Run integration tests:
  - end to end “black-box” testing of  $\mu$ ONOS 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

```
...cts/onos-test (zsh) 311 X ...cts/onos-topo (zsh) 312
→ onos-test git:(master) onit create cluster onos-test
✓ Creating cluster namespace
✓ Setting up RBAC
✓ Setting up Atomix controller
✓ Starting Raft partitions
✓ Adding secrets
✓ Bootstrapping onos-topo cluster
✓ Bootstrapping onos-config cluster
✓ Setting up GUI
✓ Setting up CLI
✓ Creating ingress for services
onos-test
→ onos-test git:(master) kubectl get ns
NAME          STATUS   AGE
default       Active  11m
kube-node-lease  Active  11m
kube-public   Active  11m
kube-system   Active  11m
onos-test     Active  47s
→ onos-test git:(master) kubectl get p
```

# ONIT Development Workflow

```
X ...cts/onos-test (zsh) 31 X ...cts/onos-topo (zsh) 312
kube-node-lease Active 11m
kube-public Active 11m
kube-system Active 11m
onos-test Active 47s
➔ onos-test git:(master) kubectl get pods -n onos-test
NAME READY STATUS RESTARTS AGE
atomix-controller-6bdd87f54f-72c92 1/1 Running 0 52s
onos-cli-6899b56c7-vpprk 1/1 Running 0 13s
onos-config-77dbf5896f-6f4sv 1/1 Running 0 28s
onos-config-envoy-58db8f7b6-swndb 1/1 Running 0 27s
onos-gui-67b867c7c7-z8pgd 1/1 Running 0 14s
onos-topo-7fdb44bb88-b5kfr 1/1 Running 0 36s
onos-topo-envoy-8899bf5d8-6dllv 1/1 Running 0 36s
raft-1-0 1/1 Running 0 45s
➔ onos-test git:(master) onit -h
Run onos integration tests on Kubernetes

Usage:
  onit [command]

Available Commands:
  add          Add resources to the cluster
  completion  Generated bash or zsh auto-completion script
  create      Create a test resource on Kubernetes
  debug      Open a debugger port to the given resource
  delete     Delete Kubernetes test resources
  fetch     Fetch resources from the cluster
  get       Get test configurations
  help     Help about any command
  onos-cli  Open onos-cli shell for executing commands
  remove   Remove resources from the cluster
  run     Run integration tests
  set     Set test configurations
  ssh    Open a ssh session to a node for executing remote commands

Flags:
  -h, --help  help for onit

Use "onit [command] --help" for more information about a command.
➔ onos-test git:(master) █
```

# ONIT Development Workflow

```
...cts/onos-test (zsh) 31 | X ...cts/onos-topo (zsh) 32 |
Run onos integration tests on Kubernetes

Usage:
  onit [command]

Available Commands:
  add          Add resources to the cluster
  completion  Generated bash or zsh auto-completion script
  create      Create a test resource on Kubernetes
  debug       Open a debugger port to the given resource
  delete      Delete Kubernetes test resources
  fetch       Fetch resources from the cluster
  get         Get test configurations
  help        Help about any command
  onos-cli    Open onos-cli shell for executing commands
  remove      Remove resources from the cluster
  run         Run integration tests
  set         Set test configurations
  ssh        Open a ssh session to a node for executing remote commands

Flags:
  -h, --help  help for onit

Use "onit [command] --help" for more information about a command.
→ onos-test git:(master) onit get tests
atomix-list
atomix-lock
atomix-map
atomix-simple
config-opstate-cli
config-plugins-cli
device-cli
device-service
ha
models
single-path
single-state
subscribe
transaction
→ onos-test git:(master) onit run test |
```

# ONIT Development Workflow

```
onit (onit) 311 X ...cts/onos-topo (zsh) 312
→ 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

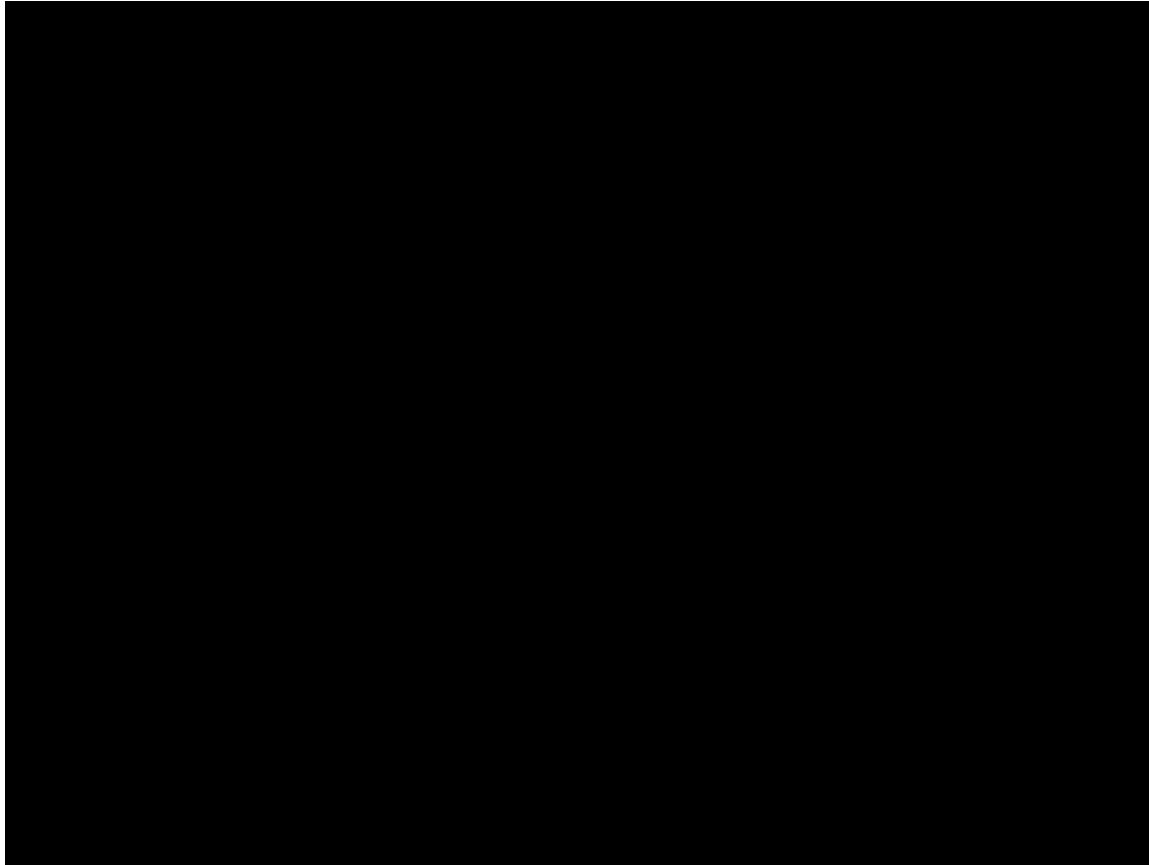
Use "onos topo get [command] --help" for more information about a command.
~ $ onos topo get devices
ID ADDRESS VERSION
~ $ onos topo add device test --address test:1234 --version
```

# ONIT Development Workflow

The screenshot displays an IDE window for a Go project named 'onos-topo'. The left sidebar shows a file explorer with a tree structure including folders like 'certs', 'cli', 'manager', and 'northbound', and files like 'device.go', 'output.go', and 'root.go'. The main editor shows the code for 'device.go', which implements a CLI command to interact with a device. The code includes logic for connecting to a device, listing its details, and printing them in a structured format. A yellow highlight is present on lines 86-87 of the code. In the top right corner, a 'Tunnellink' window shows the status of an 'ONLAB VPN' as 'Disconnected', with 'In: 0.00 B/s' and 'Out: 0.00 B/s'.

```
41 func runGetDeviceCommand(cmd #cobra.Command, args []string) {
42     t, _ := cmd.Flags().GetString("type")
43     verbose := cmd.Flags().GetBool("verbose")
44     noHeaders, _ := cmd.Flags().GetBool("no-headers")
45
46     conn := getConnection()
47     defer conn.Close()
48
49     client := device.NewDeviceServiceClient(conn)
50
51     ctx, cancel := context.WithTimeout(context.Background(), 15*time.Second)
52     defer cancel()
53     if len(args) == 0 {
54         stream, err := client.List(ctx, &device.ListRequest{})
55         if err != nil {
56             ExitWithError(ExitBadConnection, err)
57         }
58     }
59
60     writer := new(tabwriter.Writer)
61     writer.Init(os.Stdout, minwidth: 0, tabwidth: 0, padding: 3, padchar: ' ', tabwriter.FilterHTML)
62
63     if noHeaders {
64         if verbose {
65             fmt.Fprintln(writer, fmt.Sprintf("ID\tADDRESS\tVERSION\tUSER\tPASSWORD"))
66         } else {
67             fmt.Fprintln(writer, fmt.Sprintf("ID\tADDRESS\tVERSION"))
68         }
69     }
70
71     for {
72         response, err := stream.Recv()
73         if err == io.EOF {
74             break
75         } else if err != nil {
76             ExitWithError(ExitError, err)
77         }
78
79         dev := response.Device
80         if t == "" || dev.Type == device.Type(t) {
81             if verbose {
82                 fmt.Fprintln(writer, fmt.Sprintf("ID\tADDRESS\tVERSION\tUSER\tPASSWORD"))
83             } else {
84                 fmt.Fprintln(writer, fmt.Sprintf("ID\tADDRESS\tVERSION"))
85             }
86         }
87     }
88     writer.Flush()
89 } else {
90     response, err := client.Get(ctx, &device.GetRequest{
91         ID: device.ID(args[0]),
92     })
93     if err != nil {
94         ExitWithError(ExitBadConnection, err)
95     }
96
97     dev := response.Device
98
99     writer := new(tabwriter.Writer)
100    writer.Init(os.Stdout, minwidth: 0, tabwidth: 0, padding: 3, padchar: ' ', tabwriter.FilterHTML)
101    fmt.Fprintln(writer, fmt.Sprintf("ID\tADDRESS\tVERSION\tUSER\tPASSWORD"))
102    fmt.Fprintln(writer, fmt.Sprintf("ID\tADDRESS\tVERSION\tUSER\tPASSWORD"))
103    runGetDeviceCommand(cmd #cobra.Command, args []string)
```

# ONIT Development Workflow



# ONIT Development Workflow

```
...cts/onos-test (zah) 3/1 | ...cts/onos-topo (zah) 3/2 | ...cts/onos-cli (zah) 3/3
→ onos-test git:(master) onit set image onos-cli --image onosproject/onos-cli:latest
✓ Loading pod configurations
✓ Updating deployment
✓ Waiting for pods to become ready (1/1) onos-cli-66dd5c9c59-4jnhn: Pending
→ onos-test git:(master) kubectl get pods -n onos-test
NAME                                READY   STATUS    RESTARTS   AGE
atomix-controller-6bdd87f54f-72c92  1/1    Running   0           59m
onos-cli-66dd5c9c59-4jnhn           1/1    Running   0           10s
onos-cli-f4b6588c7-29fck            1/1    Terminating 0           5m33s
onos-config-77dbf5896f-6f4sv       1/1    Running   0           58m
onos-config-envoy-58db8f7b6-swndb  1/1    Running   0           58m
onos-gui-67b867c7c7-z8pgd           1/1    Running   0           58m
onos-topo-7fdb44bb88-b5kfr         1/1    Running   0           59m
onos-topo-envoy-8899bf5d8-6dllv    1/1    Running   0           59m
raft-1-0                             1/1    Running   0           59m
test-3263932912-4x5tf              0/1    Completed 0           20m
→ onos-test git:(master) |
```

# ONIT Development Workflow

```
onit (onit) 311 | X ...cts/onos-topo (zsh) 312 | X ...ects/onos-ctl (zsh) 313
→ onos-test git:(master) onit onos-ctl
~ $ onos topo get devices -h
Get a device

Usage:
  onos topo get device <id> [flags]

Aliases:
  device, devices

Flags:
  -h, --help           help for device
  --no-headers         disables output headers
  -t, --type string    get devices of the given type
  -v, --verbose        whether to print the device with verbose output
~ $ onos topo get devices -t foo
ID ADDRESS VERSION
~ $
```



# ONIT Development Workflow

```
onit (onit) 311 X ..cts/onos-topo (ssh) 312 X ..ects/onos-cl (zsh) 313
→ 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 -a -m "Add device type filter to devices command."
[device-type-filter c925414] 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: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 563 bytes | 563.00 KiB/s, done.
Total 5 (delta 4), 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/kuujo/onos-topo/pull/new/device-type-filter
remote:
To github.com:kuujo/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](#).

A screenshot of the GitHub "Open a pull request" interface. At the top, there are dropdown menus for "base repository: onosproject/onos-topo", "base: master", "head repository: kuujo/onos-topo", and "compare: device-type-filter". A green message states "Able to merge. These branches can be automatically merged." Below this is a text area with the title "Add device type filter to devices command". To the right of the text area is a "Request a review" dropdown menu. The menu is open, showing a list of reviewers: adibrastegarnia, Andrea-Campanella (highlighted), antonjin, bocan13, ccascone, onos-builder, ray-milkey, SeanCondon, tomikazi (checked), and onosproject/core. At the bottom of the text area is a green "Create pull request" button. Below the text area, it shows "1 commit" and "1 file changed". At the bottom, there is a commit history section for "Commits on Sep 10, 2019" with a commit by "kuujo" titled "Add device type filter to devices command."

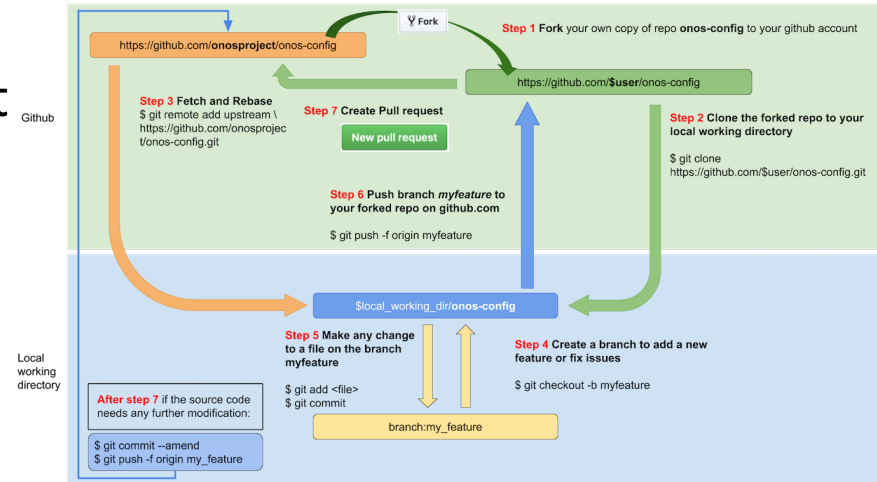
# Contributing to μONOS

Follow the developer workflow:

[https://github.com/onosproject/onos-config/blob/master/docs/dev\\_workflow.md](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



# Github

$\mu$ ONOS 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

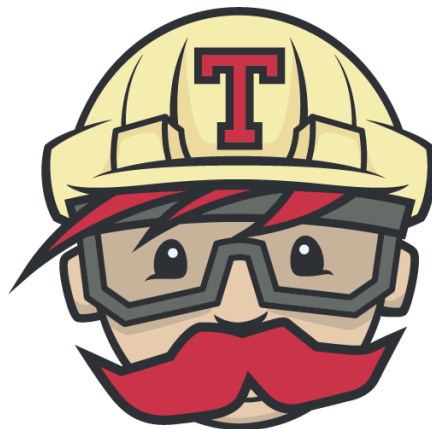


# Github and Travis

Travis, simple, well know, 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](mailto:onos-dev@onosproject.org) mailing list



# Thank You

Follow Up Links:

[μONOS repositories](#)

[Atomix repositories](#)