Open Command Line Interface Platform (OCLIP)
&
ONAP CLI

Kanagaraj.Manickam@huawei.com

CLI PTL

ONAP Paris Developer Event

25 – 09 - 2017

Agenda

- Command Line Interface (CLI)
  - What is CLI
  - Why CLI is required
  - Today’s problem in implementing CLI

- Open CLI Platform (OCLIP)
  - Open Command Specification (OCS) 1.0
  - Interactive shell
  - Web command console
  - Binaries & Installer

- ONAP CLI
  - DEMO

ONAP
OPEN NETWORK AUTOMATION PLATFORM

Command Line Interface (CLI)

What is CLI?

- Command line interface (CLI)

- Integral part of any operating system and software platforms/products
  - GUI : For Desktop vs CLI : For Console

- **Simple, Fast and Easy**

- Examples:
  - Git, svn, npm **(developers)**
  - ipconfig, hostname, apt-get, dir, ping, ssh (**operators**)
  - openstack, onap, docker, kubeadm, npm (**users**)

Why CLI is required?

- Easy automation by using scripting language (devops)
- Uniform interface
- Industry tread !
- Faster for performing operation vs GUI
- Short development cycle compare to GUI

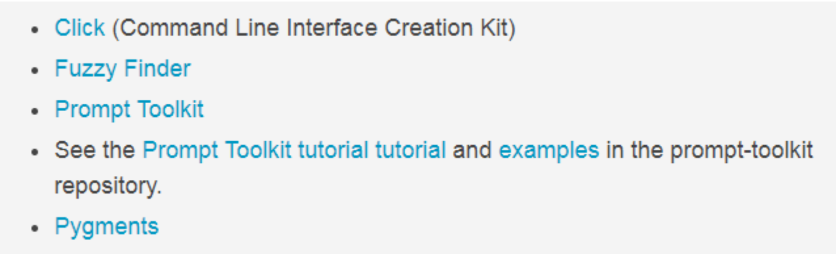
Today’s problem in implementing CLI

- There are many libraries in market
  - Dependency
  - deep learning
  - delay on getting bug-fix/feature
    - Risk.
- Community/product maintains xxx KLOC of code
  - Human-effort, money and time
  - CI (Power and energy wastage)

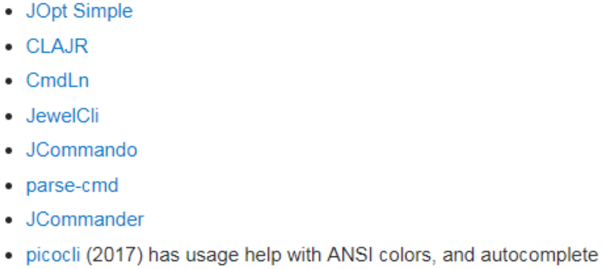
  - Delay in delivering bug-fix/feature
    - (release cycle dependency)
- Commands do not tell about output, user

Needs to run and find out ! Its issue

No platform exists today



- Click (Command Line Interface Creation Kit)
- Fuzzy Finder
- Prompt Toolkit
- See the Prompt Toolkit tutorial tutorial and examples in the prompt-toolkit repository.
- Pygments



- JOpt Simple
- CLAJR
- CmdLn
- JewelCli
- JCommando
- parse-cmd
- JCommander
- picocli (2017) has usage help with ANSI colors, and autocomplete

Platform/Community CLI	Commits	Contributors
OpenStack	4106	228
Docker	4559	557
Rancher	213	15
K8S	655	115

ONAP
OPEN NETWORK AUTOMATION PLATFORM

OPEN Command Line Interface Platform (OCLIP)

Open CLI Platform (OCLIP)

OPEN SOURCEd

Implement CLI

Only by text (YAML)

NO coding ☺ ☺ ☺

Simpler
Faster
Easier

Save Money
Save Time
Save Power
Save Energy

Industry First CLI platform !

Tomcat for HTTP, oclip for CLI

- **Console**
  - Interactive Command Shell
  - Web command console
    - Install once
      - Run commands across many products
    - Access it anywhere

Supportability:

- Any cloud (rest) enabled products
  - *Micro-service arch (w/ catalog)*
  - *Monolithic arch.*

- Non-cloud products

Command / OCS

Modeled into YAML schema Open CLI spec (OCS) 1.0

- Argument parsing & validation
  - With Short and long option (ls --help)
  - Positional arguments (ls /opt)
- Man page / Help (man ls)
- Multiple Output format (table/csv)
- Error reporting (0x1001)
- Exit code (0/1)
- Debug/Verbose logging (-d)
- Environment support ($xxx)
- Different Versions of service
- SSL support
- Authentication
- Default valuing

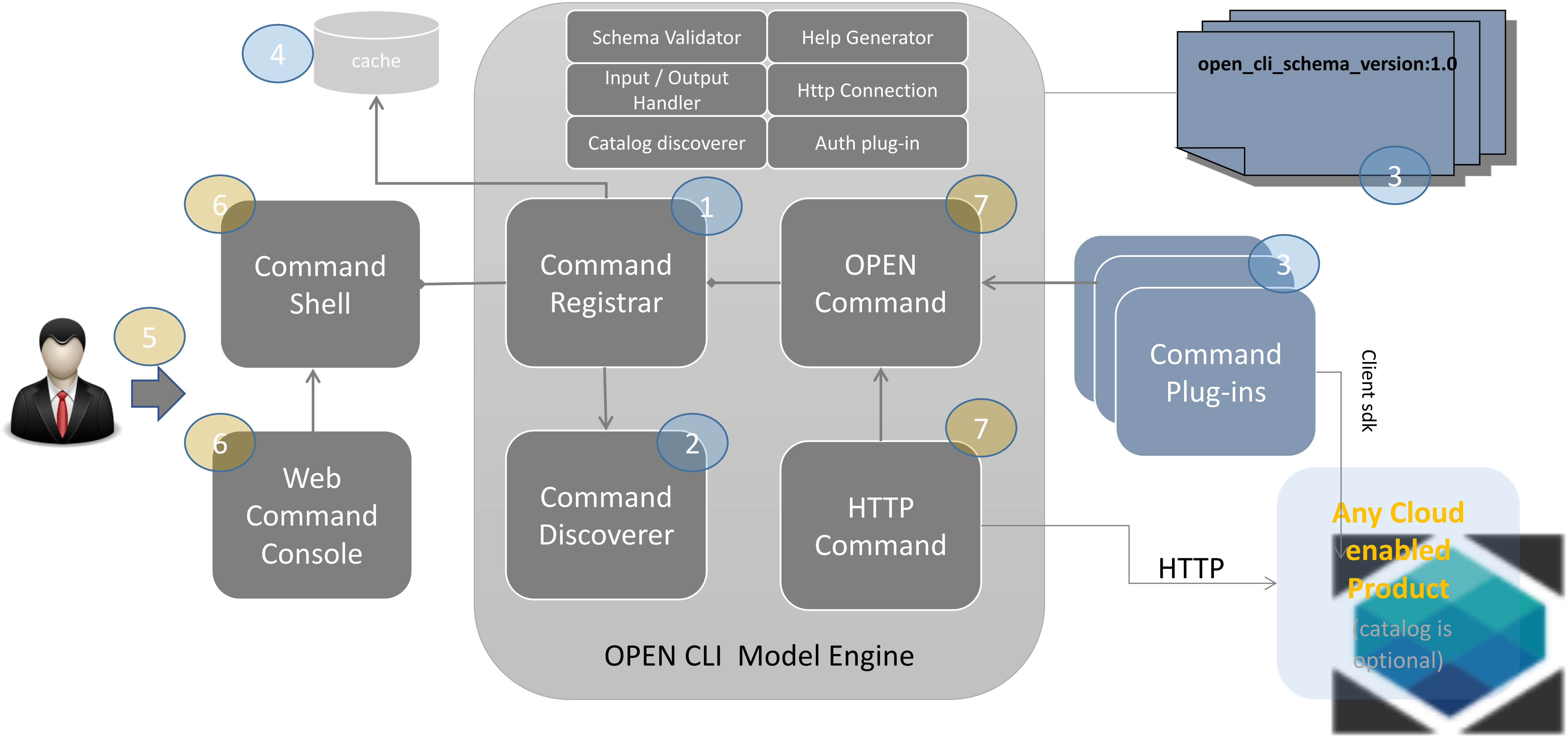
=

```
open_cli_schema_version: 1.0
name: deploy-service
description: Deploy  sample service
version: onap-1.0
parameters:
 - name: name
   description: VSP name
   type: string
   short_option: n
   long_option: name
   is_optional: false
  ......
results:
  direction: portrait
  attributes:

   - name: id
     description: service instance id
     scope: short
     type: string
   .......
```

+

```
service:
  name: vid
  version: v1
  auth:  basic/aaf
  mode:  direct/catalog
http:
  request:
    uri: /vid/rest/service
    method: POST
    body: '{"name":"${name}", ...... }‘

  result_map:
   id : $b{$.vimId}
```

OPEN CLI Platform Architecture



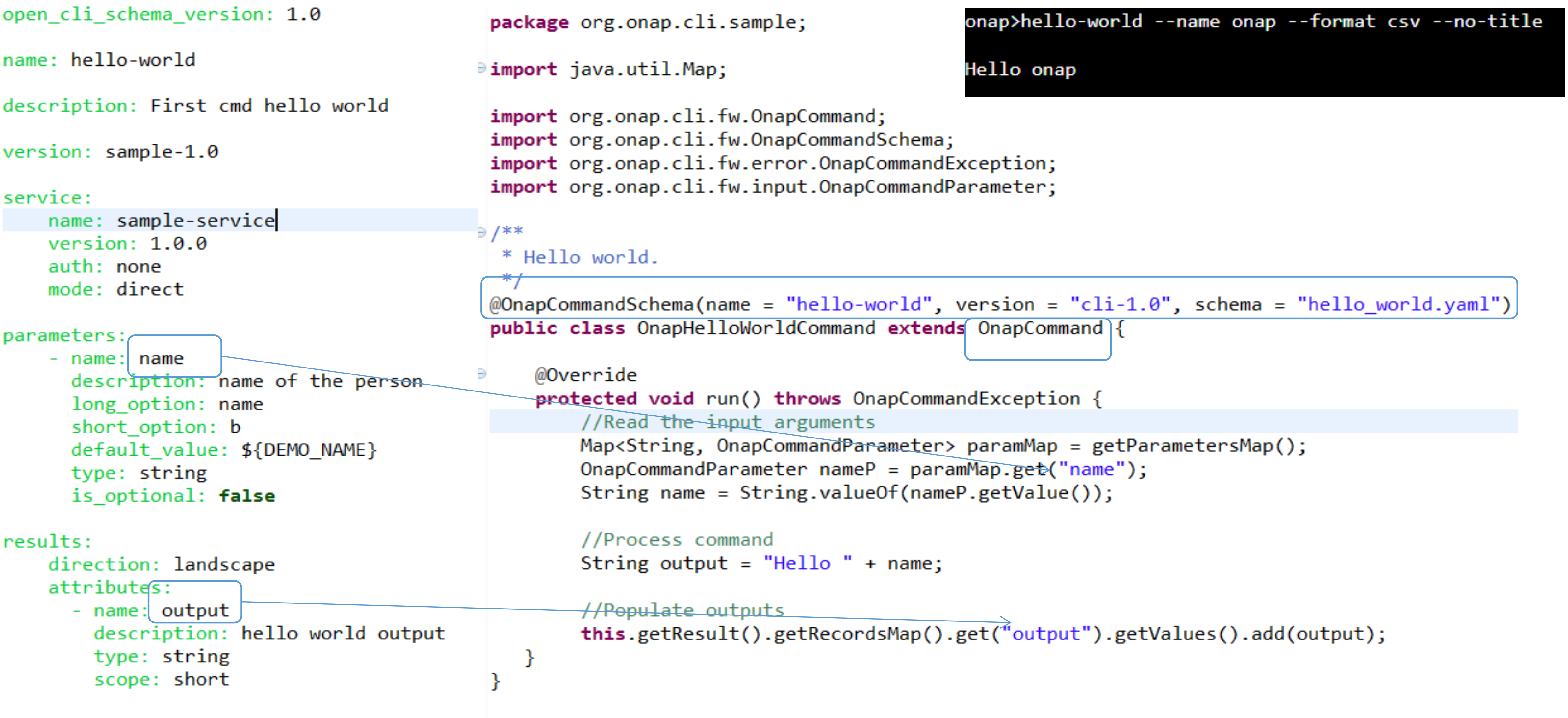
Implementing command as plug-in

- Plug-in approach is useful for implementing commands for those products does not support REST API.

- OCLIP uses this approach to provide the commands for its platform related operations and provides following commands as plug-ins:
  - **Schema-validate :** To validate the OCS YAML
  - **Schema-refresh:** To enable the newly added commands

- Offers flexibility to implement any kind of commands. For example, OCLIP provides specific plug-in command to handle HTTP commands, which made **'No code, only Text'**

Sample plug-in command : Hello world !



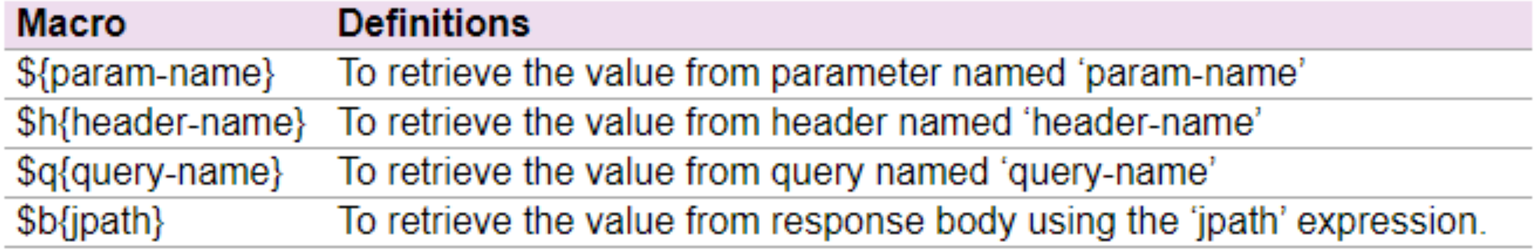
```yaml
open_cli_schema_version: 1.0

name: hello-world

description: First cmd hello world

version: sample-1.0

service:
    name: sample-service
    version: 1.0.0
    auth: none
    mode: direct

parameters:
    - name: name
      description: name of the person
      long_option: name
      short_option: b
      default_value: ${DEMO_NAME}
      type: string
      is_optional: false

results:
    direction: landscape
    attributes:
      - name: output
        description: hello world output
        type: string
        scope: short
```

```java
package org.onap.cli.sample;

import java.util.Map;

import org.onap.cli.fw.OnapCommand;
import org.onap.cli.fw.OnapCommandSchema;
import org.onap.cli.fw.error.OnapCommandException;
import org.onap.cli.fw.input.OnapCommandParameter;

/**
 * Hello world.
 */
@OnapCommandSchema(name = "hello-world", version = "cli-1.0", schema = "hello_world.yaml")
public class OnapHelloWorldCommand extends OnapCommand {

    @Override
    protected void run() throws OnapCommandException {
        //Read the input arguments
        Map<String, OnapCommandParameter> paramMap = getParametersMap();
        OnapCommandParameter nameP = paramMap.get("name");
        String name = String.valueOf(nameP.getValue());

        //Process command
        String output = "Hello " + name;

        //Populate outputs
        this.getResult().getRecordsMap().get("output").getValues().add(output);
    }
}
```

```
onap>hello-world --name onap --format csv --no-title

Hello onap
```

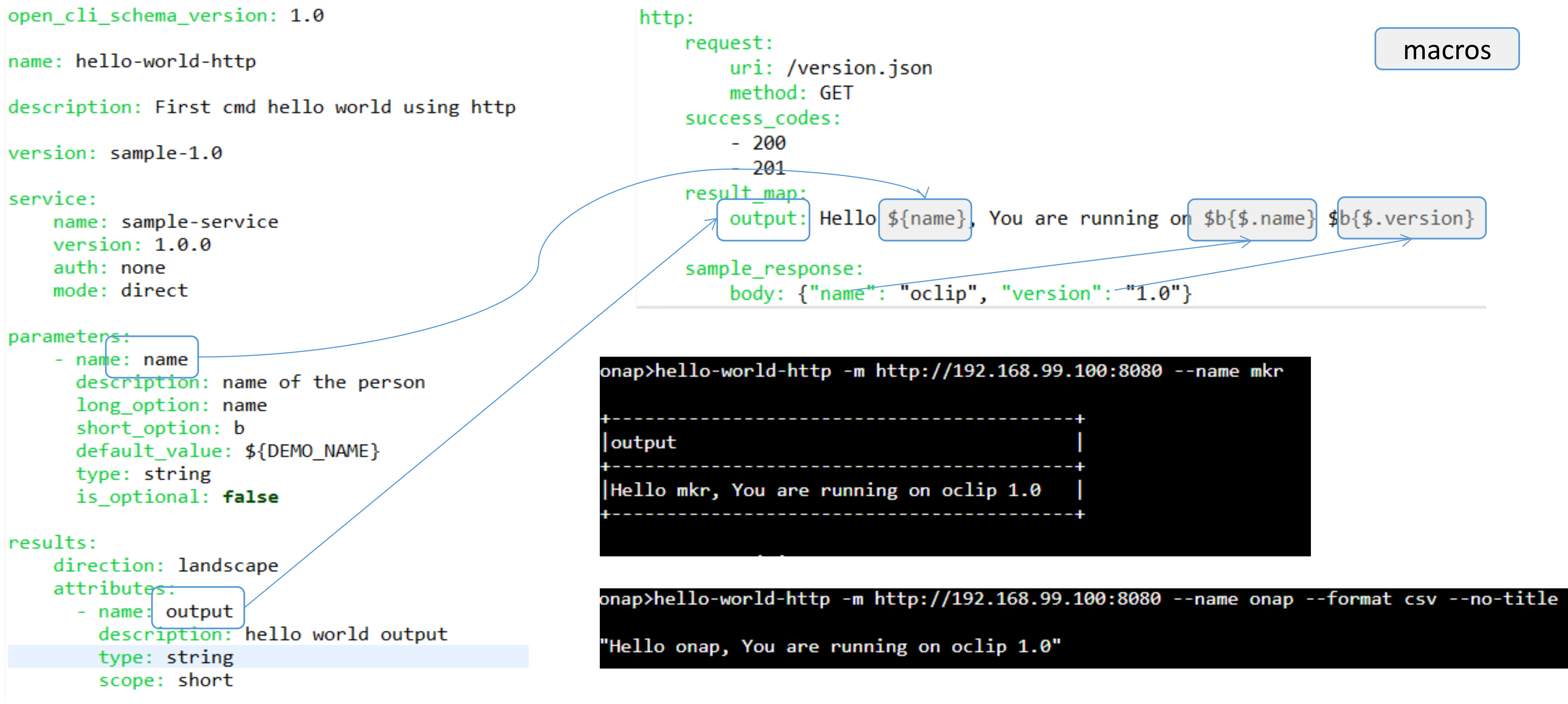
Implementing command as YAML (NO code)

- For cloud enabled products, which provides REST API, OCLIP supports to create commands just by authoring YAML file.
  - No plug-in code is required
- **http** section in OCS helps to author all HTTP action related information.
  - Uses **jpath** for processing the http response and assign the value to command results
- OCLIP provides **macros** for cross-referencing values across sections in the YAML.



Macro	Definitions
${param-name}	To retrieve the value from parameter named 'param-name'
$h{header-name}	To retrieve the value from header named 'header-name'
$q{query-name}	To retrieve the value from query named 'query-name'
$b{jpath}	To retrieve the value from response body using the 'jpath' expression.

Sample command: Hello world !



```
open_cli_schema_version: 1.0

name: hello-world-http

description: First cmd hello world using http

version: sample-1.0

service:
    name: sample-service
    version: 1.0.0
    auth: none
    mode: direct

parameters:
    - name: name
      description: name of the person
      long_option: name
      short_option: b
      default_value: ${DEMO_NAME}
      type: string
      is_optional: false

results:
    direction: landscape
    attributes:
      - name: output
        description: hello world output
        type: string
        scope: short
```

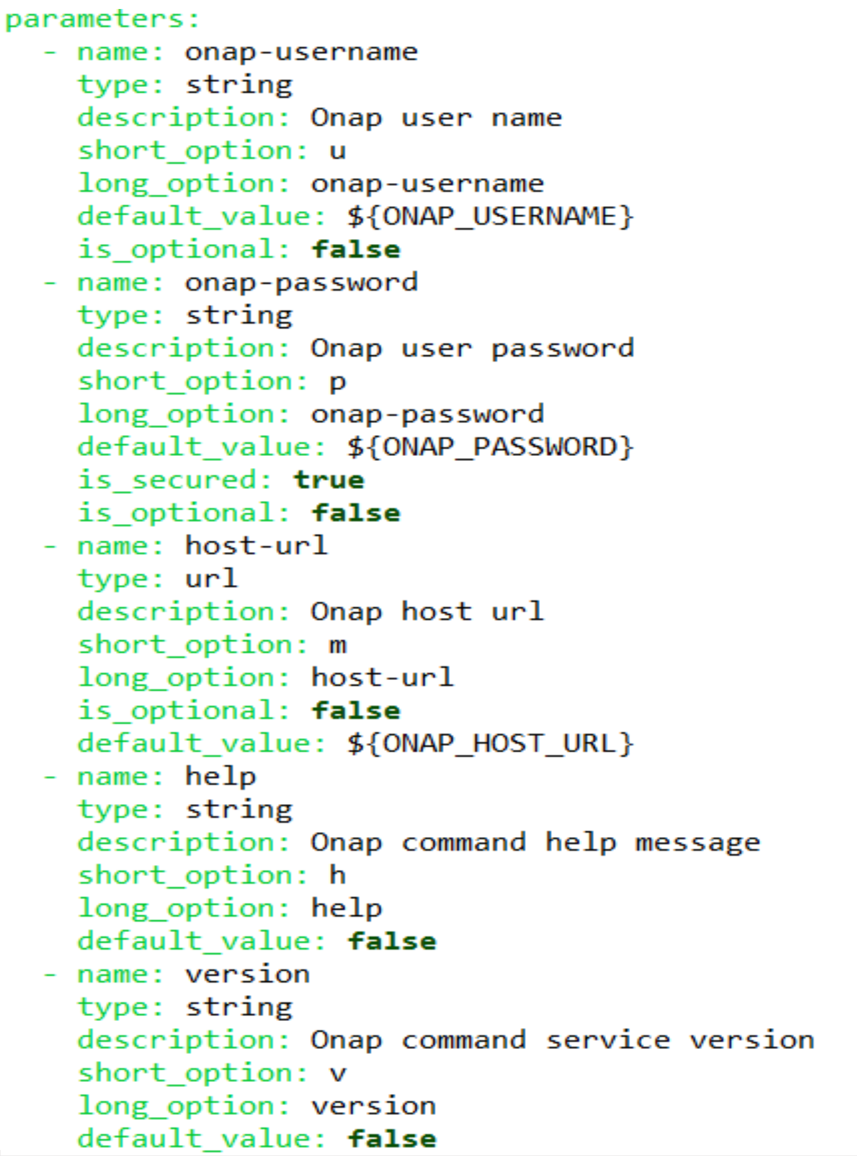
```
http:
    request:
        uri: /version.json
        method: GET
    success_codes:
        - 200
        - 201
    result_map:
        output: Hello ${name}, You are running on $b{$.name} $b{$.version}

    sample_response:
        body: {"name": "oclip", "version": "1.0"}
```

macros

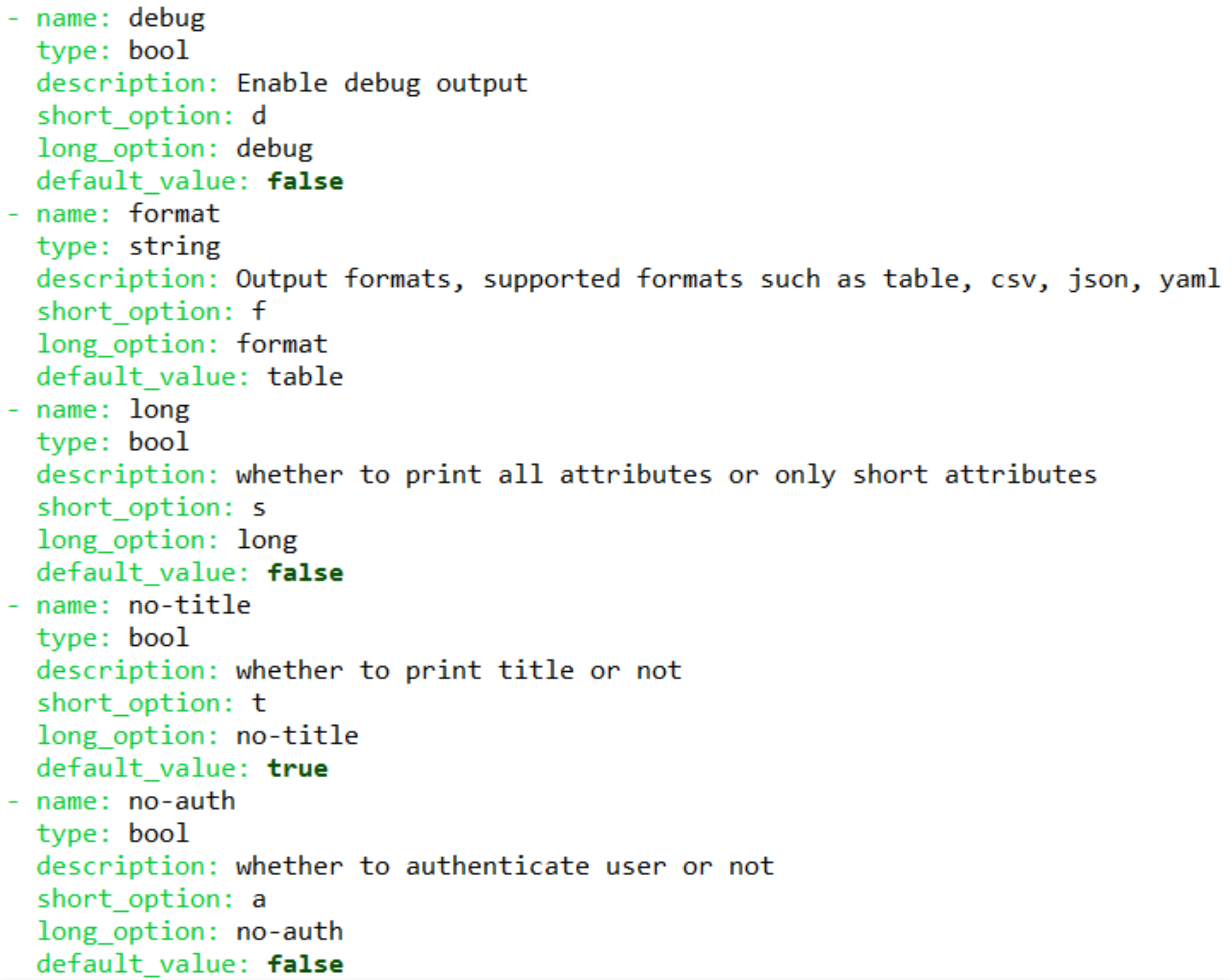
```
onap>hello-world-http -m http://192.168.99.100:8080 --name mkr

+--------------------------------------------+
|output                                      |
+--------------------------------------------+
|Hello mkr, You are running on oclip 1.0     |
+--------------------------------------------+
```

```
onap>hello-world-http -m http://192.168.99.100:8080 --name onap --format csv --no-title

"Hello onap, You are running on oclip 1.0"
```

Default Input Parameters



```
parameters:
  - name: onap-username
    type: string
    description: Onap user name
    short_option: u
    long_option: onap-username
    default_value: ${ONAP_USERNAME}
    is_optional: false
  - name: onap-password
    type: string
    description: Onap user password
    short_option: p
    long_option: onap-password
    default_value: ${ONAP_PASSWORD}
    is_secured: true
    is_optional: false
  - name: host-url
    type: url
    description: Onap host url
    short_option: m
    long_option: host-url
    is_optional: false
    default_value: ${ONAP_HOST_URL}
  - name: help
    type: string
    description: Onap command help message
    short_option: h
    long_option: help
    default_value: false
  - name: version
    type: string
    description: Onap command service version
    short_option: v
    long_option: version
    default_value: false
```



```
  - name: debug
    type: bool
    description: Enable debug output
    short_option: d
    long_option: debug
    default_value: false
  - name: format
    type: string
    description: Output formats, supported formats such as table, csv, json, yaml
    short_option: f
    long_option: format
    default_value: table
  - name: long
    type: bool
    description: whether to print all attributes or only short attributes
    short_option: s
    long_option: long
    default_value: false
  - name: no-title
    type: bool
    description: whether to print title or not
    short_option: t
    long_option: no-title
    default_value: true
  - name: no-auth
    type: bool
    description: whether to authenticate user or not
    short_option: a
    long_option: no-auth
    default_value: false
```

Authentication & Catalog

- In OCLIP, **everything is command**
  - auth and catalog are modeled as commands
  - Author needs to write these commands specific to their product/service.

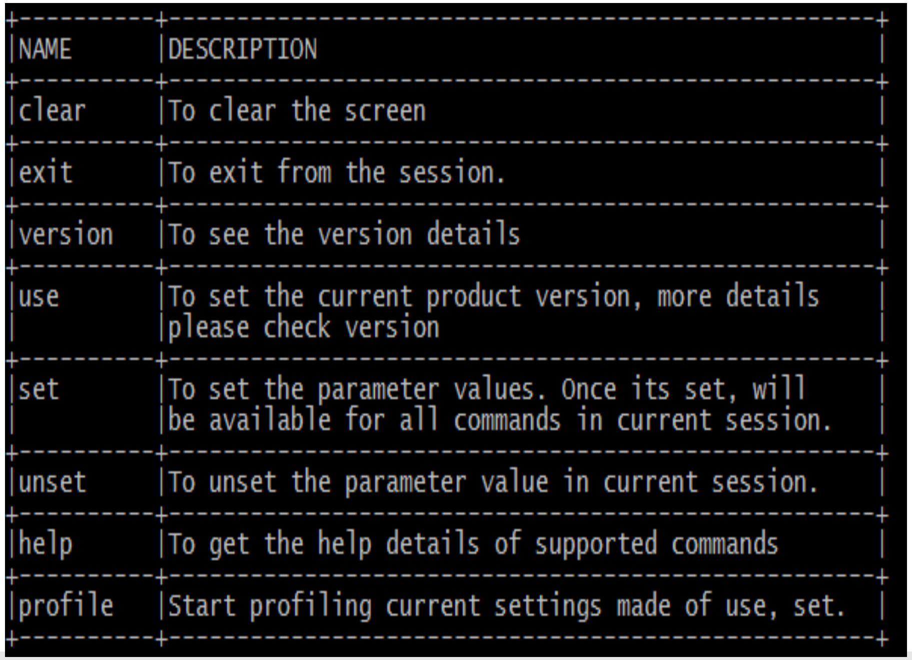
- OCLIP support matrix for product/services auth & catalog:

Product/Service	Catalog	Authentication	Features	Example
Product/Service A	YES	NO	xxx	Open-O Auth service
Product /Service B	YES	YES	yyy	All Open-O services / Future ONAP services
Product/Service C	NO	YES	zzz	AAI
Product/Service D	NO	NO	aaa	MSB

Interactive Shell

- Profiling
- environment (param storage)
- Getting help
- Clear screen
- One shell , Multiple product versions
  - Version switching

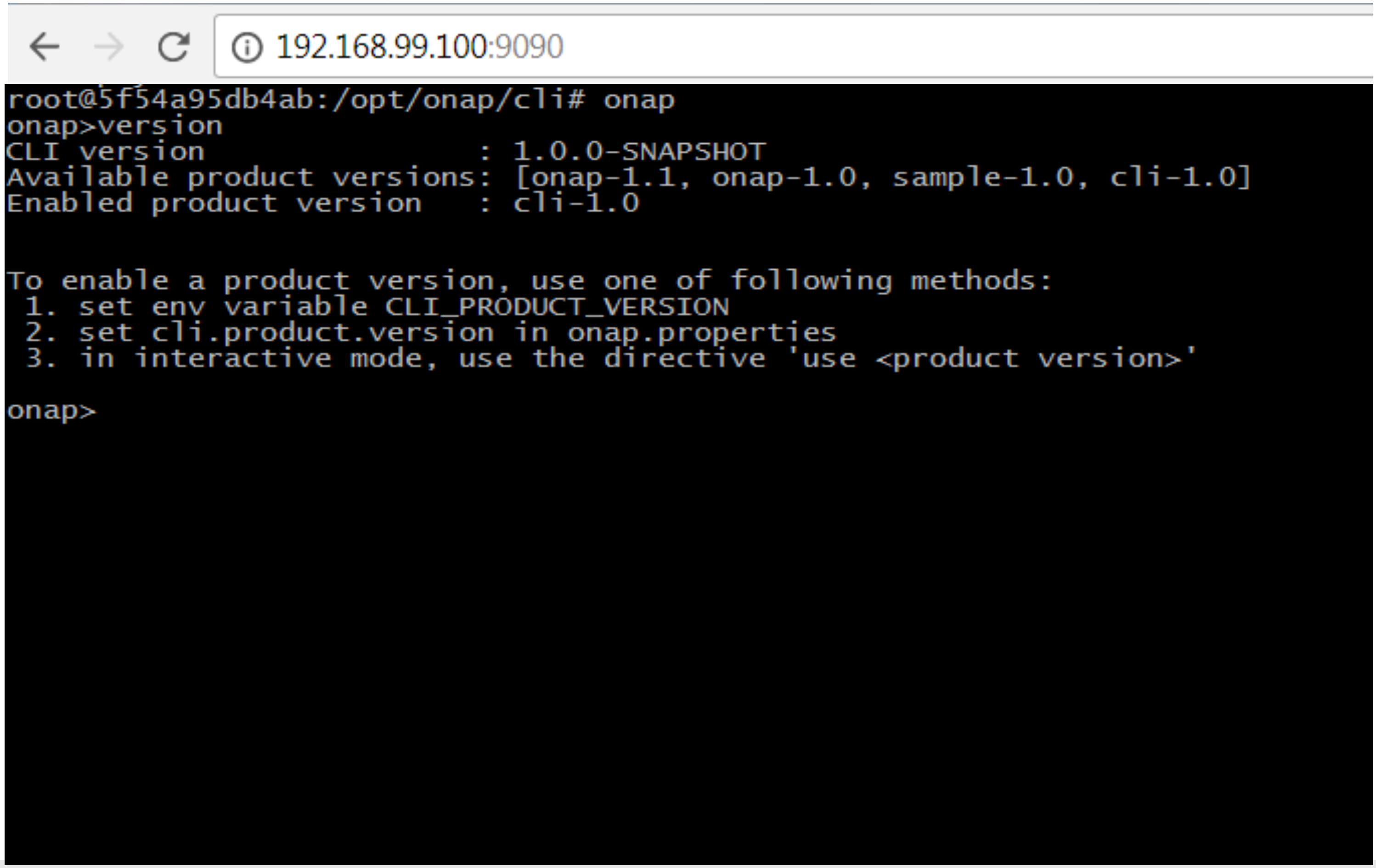
Directives



```
+----------+-----------------------------------------------+
|NAME      |DESCRIPTION                                    |
+----------+-----------------------------------------------+
|clear     |To clear the screen                            |
+----------+-----------------------------------------------+
|exit      |To exit from the session.                      |
+----------+-----------------------------------------------+
|version   |To see the version details                     |
+----------+-----------------------------------------------+
|use       |To set the current product version, more details|
|          |please check version                           |
+----------+-----------------------------------------------+
|set       |To set the parameter values. Once its set, will |
|          |be available for all commands in current session.|
+----------+-----------------------------------------------+
|unset     |To unset the parameter value in current session.|
+----------+-----------------------------------------------+
|help      |To get the help details of supported commands  |
+----------+-----------------------------------------------+
|profile   |Start profiling current settings made of use, set.|
+----------+-----------------------------------------------+
```

Command Shell

Web based Console



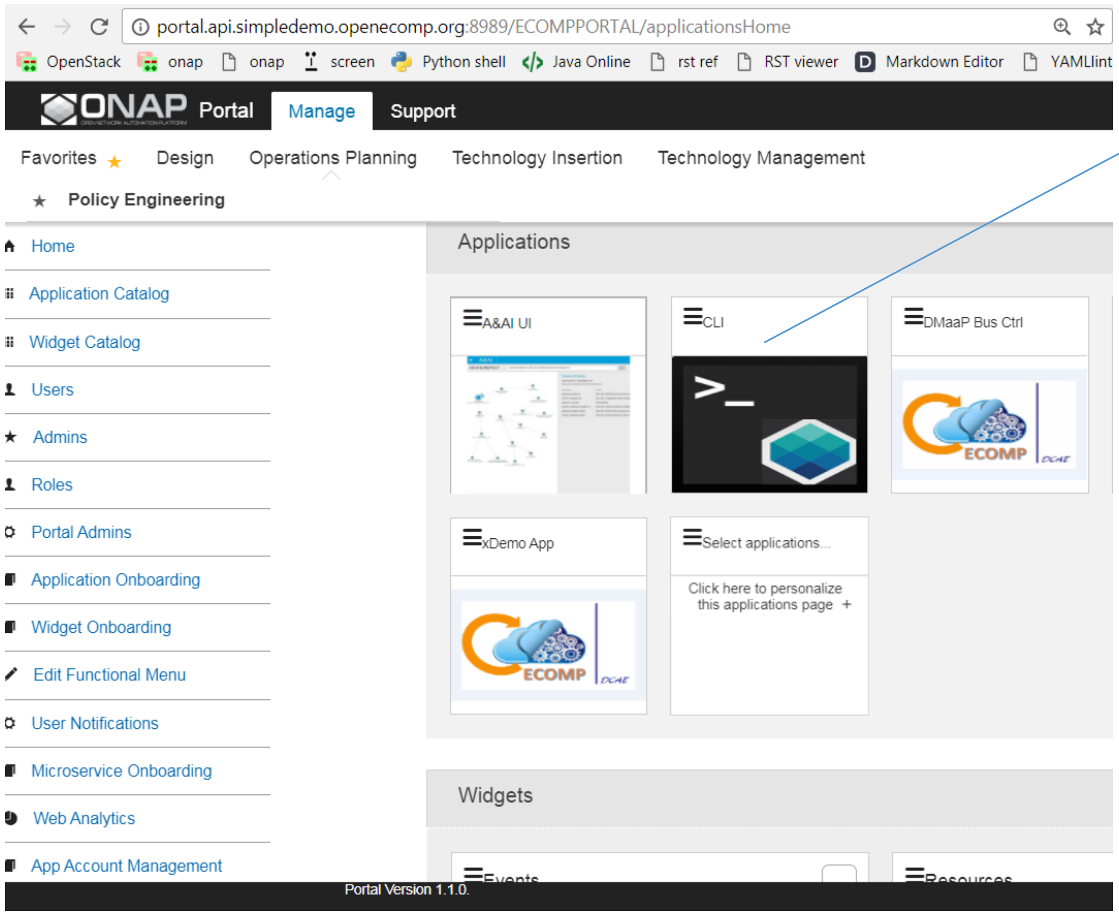
192.168.99.100:9090

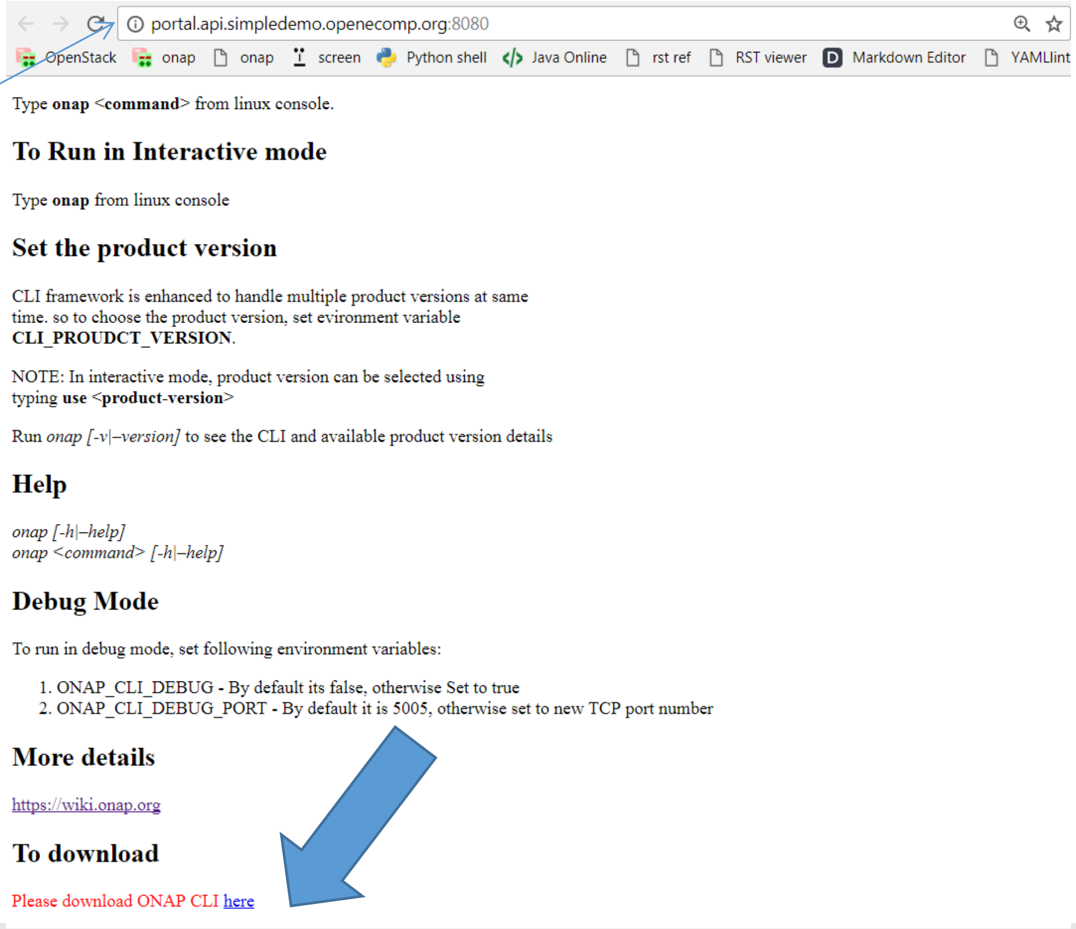
```
root@5f54a95db4ab:/opt/onap/cli# onap
onap>version
CLI version                 : 1.0.0-SNAPSHOT
Available product versions: [onap-1.1, onap-1.0, sample-1.0, cli-1.0]
Enabled product version    : cli-1.0

To enable a product version, use one of following methods:
 1. set env variable CLI_PRODUCT_VERSION
 2. set cli.product.version in onap.properties
 3. in interactive mode, use the directive 'use <product version>'

onap>
```

Binaries

Installers

(part of portal_vm)





Type **onap** <**command**> from linux console.

To Run in Interactive mode

Type **onap** from linux console

Set the product version

CLI framework is enhanced to handle multiple product versions at same time. so to choose the product version, set evironment variable **CLI_PRODUCT_VERSION**.

NOTE: In interactive mode, product version can be selected using typing **use** <**product-version**>

Run *onap [-v|–version]* to see the CLI and available product version details

Help

onap [-h|–help]
onap <command> [-h|–help]

Debug Mode

To run in debug mode, set following environment variables:

1. ONAP_CLI_DEBUG - By default its false, otherwise Set to true
2. ONAP_CLI_DEBUG_PORT - By default it is 5005, otherwise set to new TCP port number

More details

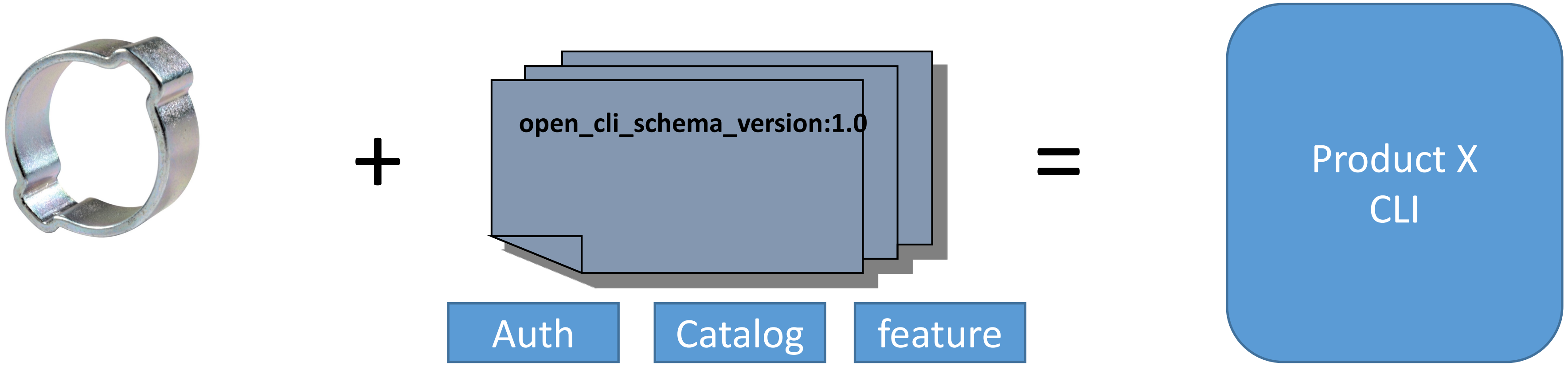
https://wiki.onap.org

To download

Please download ONAP CLI here

IT IS PLATFORM

- **Generic platform to develop CLI for any products** *(ONAP, VNF, VNFM, SDNC, EMS, your commercials, etc.)*
- Just author the required YAML files to yield CLI for any product
- To use OCLIP for a given product, author should implement following commands
  - **Authentication command**
  - **Catalog command**
  - Feature specific commands



+

open_cli_schema_version:1.0

Auth

Catalog

feature

=

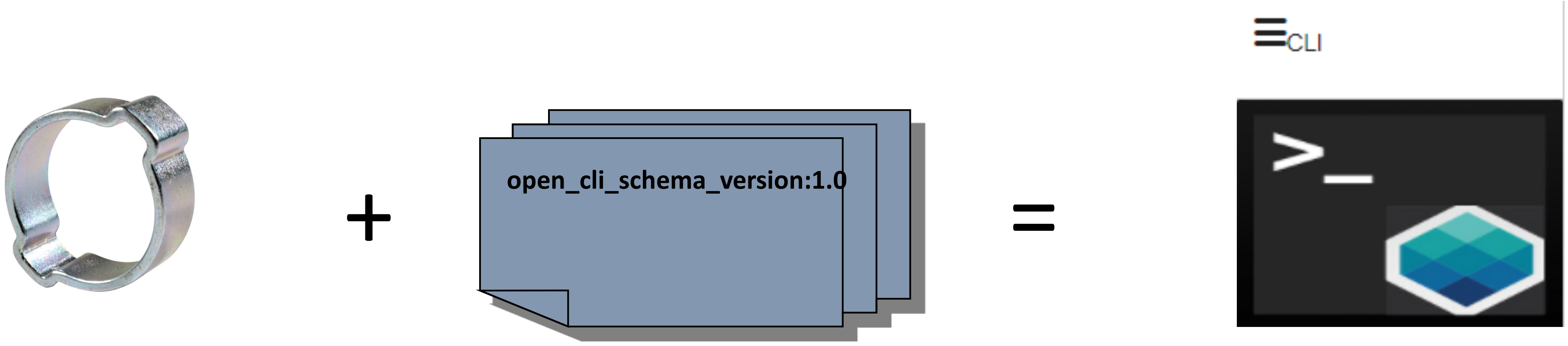
Product X CLI

ONAP CLI

ONAP CLI

- All ONAP commands are developed by authoring set of YAMLs !!
- NO Coding !!

OCLIP + Set of command YAML files -> Yields ONAP CLI



open_cli_schema_version:1.0

Auth
Basic Authentication

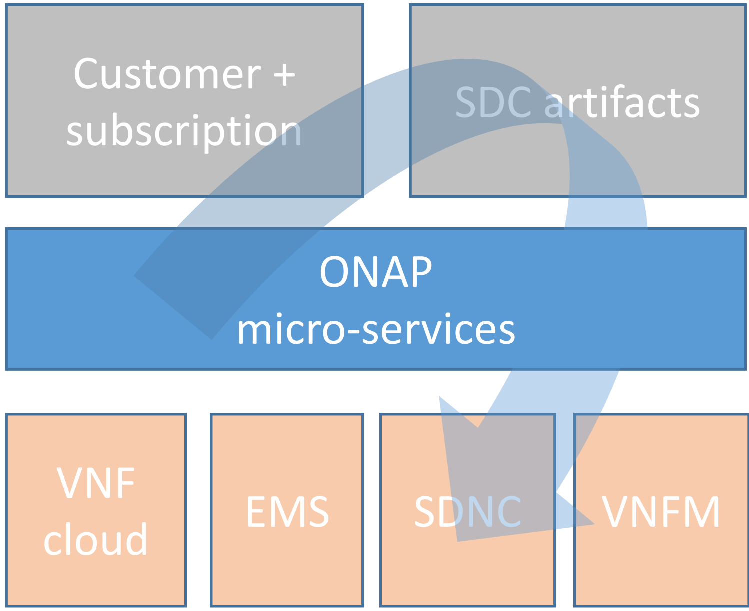
Catalog
MSB

Features
AAI, SDC, SO

ONAP CLI

Following features are supported as commands in ONAP CLI:

- ONAP micro-service discovery
- ONAP external system and VNF cloud on-boarding
- ONAP customer and subscription management
- ONAP SDC artifacts on-boarding (VF, NS TBD)
- ONAP network service life-cycle management



DEMO

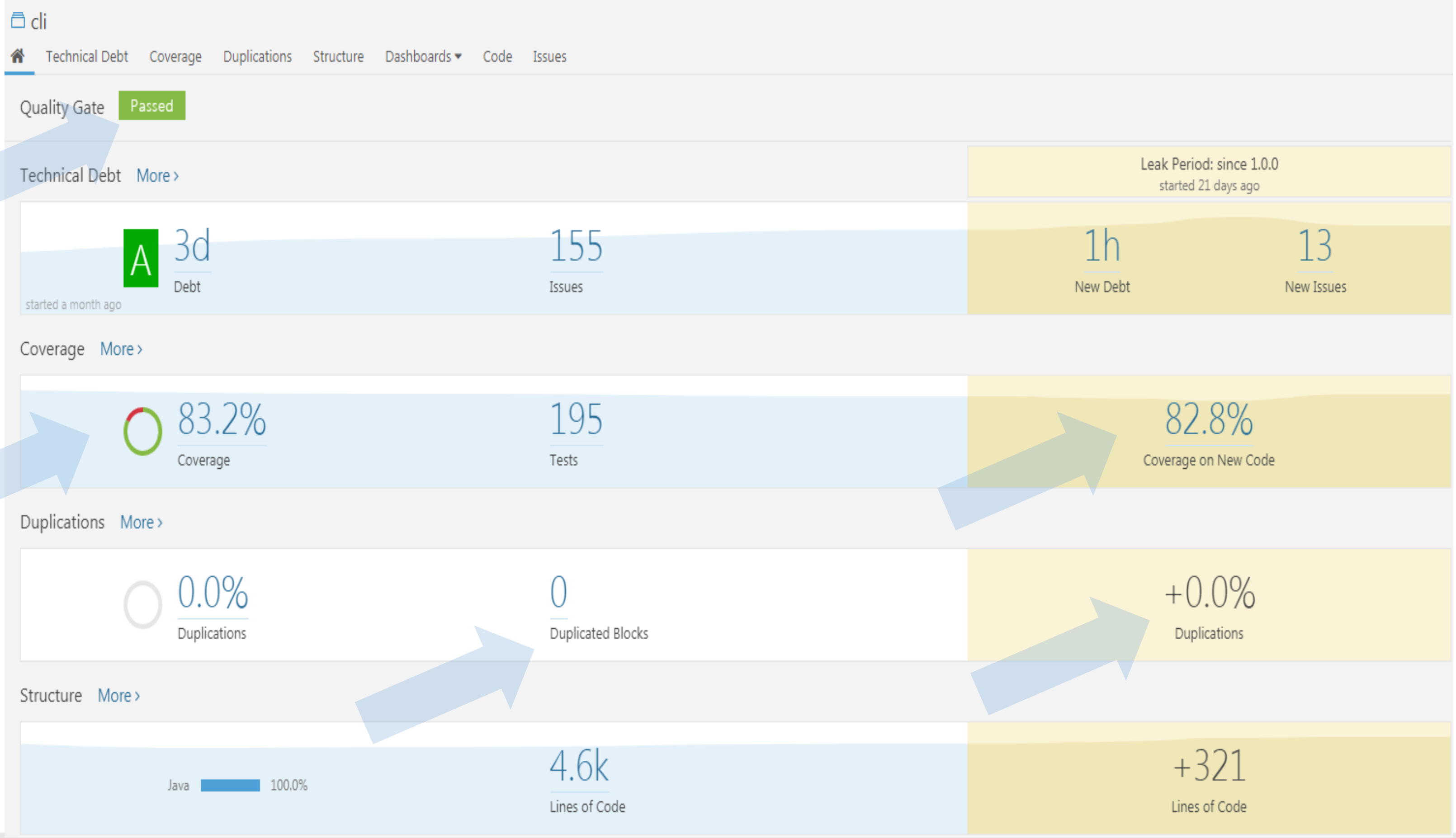
Package

- **use sample-1.0**
  - hello-world --name mkr
  - hello-world-http -m http://192.168.99.100:8080 --name mkr
- **use onap-1.1**
  - microservice-create -m http://192.168.99.100 --service-name test --service-version v1 --service-url /test/v1 192.168.99.100 8080
  - microservice-list -m http://192.168.99.100
  - microservice-show -m http://192.168.99.100 --service-name test --service-version v1

Thank you

Useful resources:

- Wiki : https://wiki.onap.org/display/DW/Command+Line+Interface+Project
- Documents: http://onap.readthedocs.io/en/latest/submodules/cli.git/docs/index.html
- Installers: Binaries & Installers
- Developer guide: TBD



cli

Quality Gate Passed

Technical Debt More ›

Leak Period: since 1.0.0
started 21 days ago

A 3d Debt
started a month ago
155 Issues
1h New Debt
13 New Issues

Coverage More ›

83.2% Coverage
195 Tests
82.8% Coverage on New Code

Duplications More ›

0.0% Duplications
0 Duplicated Blocks
+0.0% Duplications

Structure More ›

Java 100.0%
4.6k Lines of Code
+321 Lines of Code

Kanagaraj Manickam

- Sr. System Architect in Huawei
- OpenStack
  - Core-reviewer for OpenStack HEAT & TACKER
  - OpenStack Presentations
    - Heat Orchestration Template (HOT)
    - Monasca based Auto-scaling
  - Namos (OpenStack inventory controller)
- ONAP
  - CLI PTL
  - VFC committer
- Domain Experiences
  - Data-center storage, server Management & Orchestration (7 yrs)
  - OpenStack Cloud (4 yrs)
  - NFV MANO (2 yrs)
- Contact
  - Kanagaraj.manickam@huawei.com
  - IRC: KanagarajM
  - @mrkanag