Setting Up Your Development Environment

ONAP Developer Set-Up

Follow the steps below to set up the development environment on your client machine, and to establish credentials with which you may access the repositories.

- ONAP Developer Set-Up
  - Create a Linux Foundation Account
  - Basic Setup
  - Git
  - Java
  - Maven
    - Proxy settings for Maven (if required)
  - OS Specific Instructions
    - MAC/OSX (in review under 10.15.7)
    - Redhat/CentOS (in review under RHEL 8.3)
    - Ubuntu (in review under 20.04)
    - SSH Connection (Recommended)
    - HTTPS Connection
    - Proxy setting for IntelliJ and Git (if required)
  - IDEs
    - Eclipse
      - Install Eclipse
      - ONAP Eclipse Java Formatter
      - Install useful plugins for Eclipse
      - Set up Sonar towards ONAP for Eclipse
      - Setting up the ONAP Checkstyle for Eclipse
      - Spread blueprint to other projects for Eclipse
      - Configure ONAP copyright for Eclipse
    - IntelliJ
      - Download & Install
      - Configure ONAP code CheckStyle Plugin for IntelliJ
      - Configure ONAP code style auto formatting for IntelliJ (using the same checkstyle rules and automating it for you)
      - Configure ONAP copyright for IntelliJ
      - Set up SonarLint towards ONAP for IntelliJ
  - Optional Tools & Utilities
    - Python
    - git-review (optional)
    - Node-JS
    - Local SonarQube Setup
    - Other Optional Tools
  - Troubleshooting & Know Issues
    - Windows Limitations
    - Hack oparent to fix "curly bracket" issue
  - Appendix

Create a Linux Foundation Account

Follow the instructions on the identity portal to create a Linux Foundation account and get access to the ONAP Gerrit instance.

Verify that you can log in at https://gerrit.onap.org/ and that you can see the ONAP list of repositories.

Basic Setup

Git

Install git for your OS in accordance with https://www.atlassian.com/git/tutorials/install-git

Since this is common for all OS, we will also use it to generate our SSH keys as well :

```
ssh-keygen
```

This should generate a private and public ssh key.

The public ssh key can then be uploaded to Gerrit (user setting, ssh keys, add new key) or elsewhere as needed for authorization.

Java

ONAP is moving to Java 11 but many projects still use Java 8.

Download and install the appropriate openjdk version in accordance with the component you are working and set JAVA_HOME environment variable to point to that

Maven

Download the latest Maven using installer from https://maven.apache.org/download.cgi

Add maven to your path variable.

To test the new installation run:

```
mvn -version
```

Use the settings.xml found in the oparent repo

Save this Maven settings.xml as your ~/.m2/settings.xml (windows c:\users\<username>\.m2)

WSL users can point to one common repo across Windows and Ubuntu as :

```
ln -s /mnt/c/Users/<username>/.m2 ~
```

Proxy settings for Maven (if required)

If you are behind a proxy you can add a proxy section to your settings.xml

Proxy Settings

```
        <proxies>
        <proxy>
            <id>evil-corp-http</id>
            <active>true</active>
            <protocol>http</protocol>
            <host>proxy.evil-corp.com</host>
            <port>599</port>
            <nonProxyHosts>localhost|127.0.0.1|*.evil-corp.com|*.happy.evil-corp.com|fun.evil-corp.com<
/nonProxyHosts>
        </proxy>
        <proxy>
            <id>evil-corp-https</id>
            <active>true</active>
            <protocol>https</protocol>
            <host>proxy.evil-corp.com</host>
            <port>599</port>
            <nonProxyHosts>localhost|127.0.0.1|*.evil-corp.com|*.happy.evil-corp.com|fun.evil-corp.com<
/nonProxyHosts>
        </proxy>
        </proxies>
```

OS Specific Instructions

MAC/OSX (in review under 10.15.7)

get your MBP 2020 16 inch ready for development - refer to http://wiki.obrienlabs.cloud/display/DEV/Developer+Guide#DeveloperGuide-OSX

Install homebrew package manager -

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

python should already be installed. - do a *--version* check

install openstack tools

install docker (dmg) - https://docs.docker.com/docker-for-mac/install/#download-docker-for-mac

Redhat/CentOS (in review under RHEL 8.3)

Add the following option to your MAVEN_OPTS in order not to periodically hang on downloading artifacts in linux	-Djava.net.preferIPv4Stack=true
enable IP4 forwarding to enable the RHEL VM to act as an edge router - without this docker containers cannot communicate	add to /etc/sysctl.conf net.ipv4.ip_forward = 1
Install docker (required for some repos like **dcae-inventory** off dcae	`to install kubernetes follow Cloud Native Deployment#Scriptedundercloud(Helm/Kubernetes/Docker) andONAPinstall-SingleVM`

Ubuntu (in review under 20.04)

enable ubuntu desktop	apt-get update apt-get install ubuntu-desktop
enable static IP	vi /etc/network/interfaces add iface enp0s25 inet static address 192.168.15.101 netmask 255.255.255.0 network 192.168.15.0 broadcast 192.168.15.255 gateway 192.168.15.1 dns-nameservers 8.8.4.4

disable strict host checking (client and server)	`in ~/.ssh/config`
	
	`Host *`
	`  StrictHostKeyChecking no`
	`  UserKnownHostsFile=/dev/null`

Enable non CD repositories so an apt-get update will work	comment out the CD first line in /etc/apt/sources.list
enable ssh if skipped during installation so we can remote ssh	sudo apt-get install openssh-server

enable root login and gui (no sudo su -)	`sudo passwd root`
	
	sudo usermod -U root
	
	sudo reboot now
	
	sudo vi /etc/lightdm/lightdm.conf
	
	[SeatDefaults]
	
	greeter-session=unity-greeter
	
	user-session=ubuntu
	
	greeter-show-manual-login=true
	
	: enable root login
	
	`vi /etc/ssh/sshd_config`
	
	`FROM:`
	`PermitRootLogin prohibit-password`
	`TO:`
	`PermitRootLogin yes`
	
	systemctl restart sshd

Add the following option to your MAVEN_OPTS in order not to periodically hang on downloading artifacts in linux	in /etc/environment -Djava.net.preferIPv4Stack=true
Install docker (required when building docker images)	use the current as of 20180621 17.03 version in the script https://git.onap.org/logging-analytics/tree/deploy/rancher/oom_rancher_setup.sh which is `curl https://releases.rancher.com/install-docker/17.03.sh | sh`

Connecting to ONAP Gerrit

SSH Connection (Recommended)

Log in to the Gerrit host https://gerrit.onap.org/, pull down the menu under your user name (at the extreme top right of the browser window), click on "Settings":

Click your account name on the top right corner of the website and click on **Settings**.

Add the public ssh key that you created in the previous step.

HTTPS Connection

If you choose to use HTTP/HTTPS, you'll need to generate an access password. Log in to the Gerrit host https://gerrit.onap.org/, pull down the menu under your user name (at the extreme top right of the browser window), click on "Settings", and select "HTTP Password"

This password may have a limited time to live, so you might get errors like this one:

```
Problem running 'git remote update gerrit'
Fetching gerrit
fatal: unable to access 'https://USERNAME@gerrit.onap.org/r/a/mso/ ': Unknown SSL protocol error in
connection to gerrit.onap.org:443 error: Could not fetch gerrit
```

Regenerating a password will most likely solve the connectivity issue. Sometimes, the Gerrit interface on HTTPS might be temporarily faulty, so retries might be needed.

Proxy setting for IntelliJ and Git (if required)

For developers working with a proxy, you might have proxy issues connecting to Linux Foundation website. To avoid the connection issue, you must define the proxy setting for both IntelliJ and Git.

To configure the proxy settings in IntelliJ;

click **File** -> **Settings** -> **Appearance & Behavior** -> **System Settings** -> **HTTP Proxy**

enter the correct proxy settings and click **Apply**.

In case of **Manual proxy configuration**, while entering the **Host name**, there is no need to write **http://** in front of the URL. For example, if the proxy value is http://one.proxy.com, you would put **one.proxy.com** as the **Host name**

To configure the proxy settings for Git, play the following command:

```
git config --global https.proxy https://<proxy username>:<proxy password>@<proxy url>
```

```
git config --global http.proxy http://<proxy username>:<proxy password>@<proxy url>
```

Note: while entering the proxy username, sometimes it may require to add the domain name in front.

IDEs

Eclipse

Install Eclipse

Download and run the installer from: Install Eclipse. Select "Eclipse IDE for Java Developers" to install.

ONAP Eclipse Java Formatter

Download onap-java-formatter.xml and import into Eclipse.

(updates to the settings should be committed and merged back into the repository)

ONAP uses Google Java Style with some modifications. (See Java code style)

Using the Eclipse Formatter file above and maven plugin configuration in pom.xml as per pom.xml

(https://code.revelc.net/formatter-maven-plugin/ for Eclipse formatter and https://github.com/diffplug/spotless/tree/master/plugin-maven for import order)

Example pom.xml configuration

```
            <plugins>
                <!--
                Using https://code.revelc.net/formatter-maven-plugin/ for Eclipse formatter
                Using https://github.com/diffplug/spotless/tree/master/plugin-maven for import order
                Use in combination to rewrite code and imports, then checkstyle

                mvn formatter:format spotless:apply process-sources
                -->
                <plugin>
                     <groupId>net.revelc.code.formatter</groupId>
                     <artifactId>formatter-maven-plugin</artifactId>
                     <version>2.8.1</version>
                     <configuration>
                        <configFile>${project.parent.basedir}/onap-java-formatter.xml</configFile>
                     </configuration>
                     <!-- https://code.revelc.net/formatter-maven-plugin/
                          use mvn formatter:format to rewrite source files
                          use mvn formatter:validate to validate source files -->
                 </plugin>
                <plugin>
                  <groupId>com.diffplug.spotless</groupId>
                  <artifactId>spotless-maven-plugin</artifactId>
                  <version>1.18.0</version>
                  <configuration>
                    <java>
                     <importOrder>
                       <order>com,java,javax,org</order>
                     </importOrder>
                    </java>
                  </configuration>
                <!-- https://github.com/diffplug/spotless/tree/master/plugin-maven
                     use mvn spotless:apply to rewrite source files
                     use mvn spotless:check to validate source files -->
                </plugin>
            </plugins>
```

The combination can be used in a maven command to rewrite code and imports, then checkstyle audit like so

Example maven command

```
mvn formatter:format spotless:apply process-sources
```

Install useful plugins for Eclipse

Install EclEmma for code coverage and SonarLint for static code analysis.

Set up Sonar towards ONAP for Eclipse

To bind your projects to the ONAP Sonar server, follow the instructions below.

Your projects should be imported in to Eclipse before this.

1. Right click on the project and select "SonarLint -> Bind to SonarQube or SonarCloud...".
2. Select "sonarcloud" and press "Next".
3. Click "Generate Token"
4. A browser opens and you are taken to a Sonarcloud login page
5. Login with an appropriate account from the list presented, most likely your GitHub account
6. You are now directed to a sonarcloud token generation page
7. Enter a name for your token and click "Generate"
8. Copy the token hex string that is generated from the browser and paste it into the "Token" field in Eclipse and click "Next"
9. In the "Organization" field, enter the string "onap" and press "Next"
10. The Connection name "SonarCloud/onap" should be found by the system, click "Next"
11. The connection should be successfully created, click "Finish"
12. Press "Add...".

13. Select the projects you want to add and press "OK".
14. Press "Next".
15. Start typing the name of your project, and it should appear in a list box where it should be selected.
16. Press "Finish".

To see messages from Sonar introduced by edits made in the projects, select "Window -> Show View -> Other...". Expand "SonarLint" and select "SonarLint -> On-The-Fly".

Setting up the ONAP Checkstyle for Eclipse

Set "ONAP" configuration in Eclipse

To set the newly built checkstyle files in Eclipse:

1. Preferences->Checkstyle
2. Click "New"
3. Select "External Configuration File"
4. Give it a name eg ONAP
5. Point at the file <your_git_folder>/oparent/checkstyle/src/main/resources/onap-checkstyle/onap-java-style.xml (assuming you have downloaded the *oparent* repo)
6. Click OK
7. Select "ONAP" configuration and click "Set as Default"
8. Select "Apply and Close"

Apply "ONAP" configuration to a project in Eclipse

Now we need to activate the checkstyle on one project and set it as the blueprint for all of them:

1. Select a project in eclipse and right click->PropertiesCheckstyle
2. check "Checkstyle active for this project"
3. Select the "ONAP" checkstyle profile
4. Click "Apply and Close"

Spread *blueprint* to other projects for Eclipse

Now spread the profile to all other projects:

1. Select all the projects you want to apply the profile to in the Eclipse project explorer (not the one that you set up above)
2. Right click->Checkstyle->Configure projects from blueprint
3. Select the project you set up above
4. Now all the projects have the correct checkstyle setup.

Configure ONAP copyright for Eclipse

1. Window > Preferences
2. Java > Code Style > Code Templates
3. Click Code and select New Java files
4. Click Edit
5. Paste the license below
6. Click Insert Variable and add the date
7. Add the organization name
8. Click Apply

```
============LICENSE_START=======================================================
 Copyright (C) $today.year <organization name>
 ================================================================================
 Licensed under the Apache License, Version 2.0 (the "License");
 you may not use this file except in compliance with the License.
 You may obtain a copy of the License at

       http://www.apache.org/licenses/LICENSE-2.0
 Unless required by applicable law or agreed to in writing, software
 distributed under the License is distributed on an "AS IS" BASIS,
 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 See the License for the specific language governing permissions and
 limitations under the License.

 SPDX-License-Identifier: Apache-2.0
 ============LICENSE_END=========================================================
```

IntelliJ

Download & Install

see https://www.jetbrains.com/idea/download/#section=windows

Make sure you have added the Checkstyle plugin . Follow the below link

https://plugins.jetbrains.com/plugin/1065-checkstyle-idea

Configure ONAP code CheckStyle Plugin for IntelliJ

1. Select, File, Settings, Tools, Checkstyle
2. Click on + beside the 'Configuration File' box to add a configuration
3. Set description to something like 'ONAP Rules'
4. Click on Browse to select the file <your_git_folder>/oparent/checkstyle/src/main/resources/onap-checkstyle/onap-java-style.xml (assuming you have downloaded the *oparent* repo)
5. Complete the Wizard (you can set exclusion properties if needed)
6. Select the Configuration File you just added by selecting the relevant checkbox
7. Click [OK] to close the settings popup

Configure ONAP code style auto formatting for IntelliJ (using the same checkstyle rules and automating it for you 🙂)

1. Select, File, Settings, Editor, Code Style
2. Click on the gear icon at the end of the line for "Scheme:"
3. Optional: As importing a schema overrides the current scheme you might want to first use the 'Duplicate..' and 'Rename...' options to create an easily identifiable scheme e.g. 'ONAP Standard'
4. Import SchemeCheckstyle Configuration
5. Click on Browse to select the file <your_git_folder>/oparent/checkstyle/src/main/resources/onap-checkstyle/onap-java-style.xml (assuming you have downloaded the *oparent* repo)
6. Click OK

Configure ONAP copyright for IntelliJ

1. File > Settings > Editor > Copyright > Copyright Profiles
2. Click the + icon and add the copyright text below
3. Change the organization name
4. Go to the Copyright
5. Select the new copyright profile from the drop down for Default project copyright
6. Click Apply

```
============LICENSE_START=======================================================
 Copyright (C) $today.year <organization name>
 ================================================================================
 Licensed under the Apache License, Version 2.0 (the "License");
 you may not use this file except in compliance with the License.
 You may obtain a copy of the License at

       http://www.apache.org/licenses/LICENSE-2.0
 Unless required by applicable law or agreed to in writing, software
 distributed under the License is distributed on an "AS IS" BASIS,
 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 See the License for the specific language governing permissions and
 limitations under the License.

 SPDX-License-Identifier: Apache-2.0
 ============LICENSE_END=========================================================
```

Set up SonarLint towards ONAP for IntelliJ

Prerequisites

- You need to be part of the onap organization in GitHub (see GitHub, click on your icon and select 'My Organizations')
  If you are not a member of this organization email support.linuxfoundation.org and ask to be added to the onap organization in GitHub
  Note. It is possible to enter a different organization if you at least member of 1 organization (you can create your own in GitHub).
  The plugin wizard crashes if you are member of no organization at all, see this Bug: https://jira.sonarsource.com/browse/SLI-426
- Make sure you have added the SonarLint plugin

To use SonarLint with Onap projects you need to setup a connection with SonarCloud

1. Select, File, Settings, Tools, SonarLint.
2. Click on + beside the 'SonarCube / SonarClod connections' box to add a new connection
3. Set Configuration Name to something like 'ONAP'
4. Ensure **sonarcloud** is selected and click [Next]
5. Click [Create Token]
6. Select GitHub, logon and follow the GitHub wizard to create a Token
7. Copy Token into IntelliJ wizard
8. Click [Next] (might have to wait a little while your data is being downloaded)
9. Select 'Open Network Automation Platform (ONAP)' from 'Your Organizations' and click [Next]
10. Click [Finish] to close the wizard

11. Click [OK] to leave settings

Then you need to bind each project to the corresponding project in GitHub/SonarCloud:

1. Select, File, Settings, Tools, SonarLint, Project Settings
2. Enable (check) 'Bind project to SonarQube / SonarCloud'
3. Select the connection created above using the [Configure the connection] button
4. Enter the project name (e.g. 'onap_ccsdk-sli-plugins') or select it using [Search in list..]
5. Start typing the name of your project, and it should appear in a list box where it should be selected.
6. Click [OK] to finish

Optional Tools & Utilities

Python

> Check which version of python your project is using before installing.

Download Python from https://www.python.org/downloads/

Add python binary to Path.

Install "pip3", if you plan to use git-review.

```
sudo apt-get update
sudo apt-get -y install python3-pip
pip3 --version
```

git-review (optional)

In a shell, type the following command (assumes you installed python above) :

```
pip3 install git-review
```

To use `git review`, you have to be in a Git clone directory that already contains a (possibly hidden) `.gitreview` configuration file (see Gerrit/Advanced usage#Setting up a repository for git-remote)

> The Git and git-review installation steps above are derived from the description at: https://www.mediawiki.org/wiki/Gerrit/git-review#Windows

Configure Git to remember your Linux Foundation user name and email address (the user name and email address associated with your Linux Foundation login):

```
git config --global user.email <your_LF_account_email>

git config --global --add gitreview.username <your_LF_user_name>
```

Configure git review

```
git config --global gitreview.remote origin
```

If you are using VPN, you might encounter a proxy problem while connecting to the Linux Foundation website. To avoid the problem, you should add the proxy setting in git config, using the following command:

```
git config --global https.proxy https://<proxy username>:<proxy password>@<proxy url>

git config --global http.proxy http://<proxy username>:<proxy password>@<proxy url>
```

NOTE: When entering the proxy username, you might be required to add the domain name in front of the username.

Node-JS

The dcae build will install npm in most cases.

However you can install it yourself from https://nodejs.org/en/download/

Verify your installation by

```
npm -v
```

Local SonarQube Setup

It can be useful to have SonarQube running locally rather than pushing through CI for feedback.

This can be done using docker:

Images of the Community, Developer, and Enterprise Editions are available on Docker Hub.

1. Start the server by running:

   ```
   $ docker run -d --name sonarqube -p 9000:9000 sonarqube:latest
   ```

2. Log in to http://localhost:9000 with System Administrator credentials (login=admin, password=admin).
3. Click the **Create new project** button to analyze your first project.

Other Optional Tools

The component you are working on may require additional tool installations, see the relevant section under Development Guides.

Troubleshooting & Know Issues

Windows Limitations

Some repos might not clone in Windows until some file paths are reduced below 255 chars.

Enable long paths in windows as

```
git config --system core.longpaths true
```

Hack oparent to fix "curly bracket" issue

The first issue is that the current ONAP master tagged version of the checkstyle does not work with Eclipse Oxygen/Photon (and maybe other versions) because of the "curly bracket" issue. There is a fix on the way but it's not here yet. The current tagged version of oparent we are using in Policy is 1.2.1.

To get around this issue, check out oparent, checked out the 1.2.1 tag, fixed the "curly bracket" bug and built it on my local machine.

1. Check out oparent
2. git tag -l
3. git co tags/1.2.3 -b 1.2.3
4. If you want CheckStyle to ignore generated files, do the following:
   a. Add the following tag to the above file:
      <module name="SuppressionFilter">
        <property name="file" value="<absolute path to the directory of the file>/suppressions.xml"/>
      </module>
   b. Create a file called "suppressions.xml" in the folder given above, and put the following content in it:
      <?xml version="1.0"?>
      <!DOCTYPE suppressions PUBLIC
        "-//Checkstyle//DTD SuppressionFilter Configuration 1.2//EN"
        "https://checkstyle.org/dtds/suppressions_1_2.dtd">
      <suppressions>

```
    <suppress files="[\\/]generated-sources[\\/]" checks="[a-zA-Z0-9]*"/>
  </suppressions>
```

5. The versions in the POMs in oparent are snapshot, so we need to change those to 1.2.1
   mvn versions:set -DnewVersion=1.2.1
6. Now build locally:
   mvn clean install

Appendix

- Building Entire ONAP