Build-Ship-Run approach for a CORD-in-a-Box deployment

Canellas, Ferran; Bonjorn, Nestor; Kentis, Angelos Mimidis; Soler, José

Published in:
Proceedings of 9th International Conference on Cloud Computing and Services Science

Link to article, DOI:
10.5220/0007685402870291

Publication date:
2019

Document Version
Peer reviewed version

Build-Ship-Run approach for a CORD-in-a-Box deployment

Ferran Canellas[1], Nestor Bonjorn[1], Angelos Mimidis and Jose Soler[1]
[1]*Denmarks Technical University, Building 343, Ørsteds Pl., Lyngby, Denmark*
agmimi@fotonik.dtu.dk

Keywords: CORD, CiaB, Cloudification

Abstract: 5G is expected to provide high bandwidth and low latency communications, thus allowing Telco operators to provide new services to their end customers. This increase in performance is achieved through the migration of network functions from the core to the edge of the network and facilitated by the flexibility and automation provided by Software Defined Networking (SDN) and Network Function Virtualization (NFV). To pave the way to 5G, and simplify the management of 5G deployments a number of SDN/VNF platforms has been developed in the recent years. However deploying and configuring the platforms themselves, is a complex and time consuming task which can act as a barrier to their adoption by Telco operators. This is because Telco Operators strife for fast provisioning times and zero-touch provisioning. Based on this observation, this paper proposes a Build-Ship-Run platform deployment using Central Office Re-architected as a Datacenter (CORD) as an exemplar platform. The proposed approach is based on the use of compressed Virtual Machine snapshots, which allow preconfigured CORD-flavors to be fetched, uncompressed and deployed on demand. Using the proposed workflow, a deployment time seven times better than the raw installation is demonstrated.

1 INTRODUCTION

One of the most prominent Telco-oriented, 5G-edge platforms is the Central Office Re-architected as a Datacenter (CORD) (Peterson, 2016). CORD's mission is to provide a virtualized Central Office (CO) for Telco Operators, by utilizing the Network Function Virtualization (NFV) and Software Defined Networking (SDN) paradigms. This platform can be deployed in two ways, using either a physical or a virtual deployment. A physical deployment is often referred as a POD and consists of a set of physical servers and switches. The virtual deployment, which is usually referred as CORD-in-a-Box (CiaB) is the equivalent of a POD, where the servers and switches run in a single physical host as Vagrant virtual machines (VMs) and Open vSwitch (OvS) switches. The focus throughout the work presented in this paper has been given in the CiaB deployment of the CORD 4.1 version.

CiaB is, by default, deployed by means of executing a set of scripts and *makefiles* that take care of priming the target server and downloading, installing and configuring the virtual machines that comprise CORD. However, this process is very time consuming, since it can take up to two hours to complete, depending on the target server. Moreover, bug-fixes or changes are often pushed to "stable" CORD versions, meaning that the CORD code-base might change from deployment to deployment causing loss of control over the installation base.

To address this issue, this paper proposes a new way to deploy CORD based on VM images and configuration files instead of external repositories. This new approach is named Build-Ship-Run (BSR) and is based on InstaCORD (InstaCORD, 2017), which proposed a way to export CORD VMs of a CiaB 3.0 deployment. The core idea is to use images of the VMs of a running and verified CORD deployment and store them on an online or local repository. These VM images, together with any related configuration files and execution scripts will later be used to bring up a CiaB onto a bare-metal (Linux-based) server. This approach (1) provides faster deployment times and (2) ensures that the deployed CORD components are compliant with a reference installation.

The rest of this paper is organized as follows. Section 2 describes the "Build" process, i.e., how to create a CORD backup, including all the necessary configuration files. Section 3 describes the "Ship" process, i.e., the different ways to move the CORD

backup into the target server. Section 4 describes the "Run" process, i.e., how to fire up CORD based on the output of the "Build" process. Section V presents results comparing the raw installation of CORD with different versions of the proposed Build-Ship-Run approach. Finally, Section 5 provides the conclusions of this work.

2 BUILD

The purpose of the build process is to create a backup of the raw state of a CiaB deployment, consisting of a copy of the CORD source tree, the VM images as well as several configuration files. Since the overall size of these files is around 60 GB, they are compressed in order to facilitate shipping. Thus, the build process is organized in the following steps:

1) Get network configuration files: a CiaB deployment defines four virtual networks, which interconnect the virtual machines with the virtual switches, namely cord0, cordmgmt, default and vagrant-libvirt. In this step, an XML file that contains the necessary information is generated for each of these networks.

2) Get VM configuration files: a CiaB deployment contains several VMs, namely head1, corddev and a set of compute nodes. In this step, an XML file that contains the necessary information is generated for each of these VMs.

3) Bring down the VMs and stop the virtualization service: all the running Vagrant VMs are stopped as well as the virtualization service that manages them.

4) Compression: both the CORD source tree and the VMs are compressed.

The outcome of this script is a folder that contains all the aforementioned files. This folder can be saved either in an external hard drive or in the cloud to be used for a fast deployment of CORD.

3 SHIP

As mentioned previously, there may be two different scenarios of shipping CORD: using a local repository or a remote repository. In this section, the fastest way of shipping CORD is analyzed considering both scenarios.

3.1 Local repository

Depending on the available bandwidth the optimal solution will be either to first transfer the files and then decompress them, or to transfer and decompress them at the same time. Equation (1) shows the optimal ship time (t_{ship}) depending on the available bandwidth (BW) assuming that the CORD backup is compressed, where TS is the total size of the backup, CS is the compressed size of the backup, DS is the decompression speed and CR is the compression ratio, which is defined as the ratio between the uncompressed and compressed size.

$$\mathrm{t_{ship}} = \begin{cases} \dfrac{\mathrm{CS}}{\mathrm{BW}} + \dfrac{\mathrm{TS}}{\mathrm{DS}}, \mathrm{BW} < \left(1 - \dfrac{1}{\mathrm{CR}}\right) * \mathrm{DS} \\ \dfrac{\mathrm{TS}}{\mathrm{BW}}, \left(1 - \dfrac{1}{\mathrm{CR}}\right) * \mathrm{DS} \leq \mathrm{BW} \leq \mathrm{DS} \\ \dfrac{\mathrm{TS}}{\mathrm{DS}}, \mathrm{BW} > \mathrm{DS} \end{cases} \quad (1)$$

In the first interval, the bandwidth is so small that the optimal ship time is given by transferring first the compressed backup to the target server and then decompressing it. In the other intervals the optimal time is given by decompressing and transferring the backup altogether. In the second interval, the bottleneck is in the bandwidth and, thus, t_{ship} depends on BW, whereas in the third interval, the bottleneck is in the decompression speed, and, thus, t_{ship} depends on DS. In case the CORD backup was not compressed, the time in the third interval would be smaller, specifically TS divided by BW. However, this scenario is discarded because of storage limitations since the total size of the backup would be much larger (around 57 GB) than by compressing it (around 27 GB).

3.2 Remote repository

If the backup is downloaded from an online repository, the backup cannot be downloaded and decompressed at the same time. Therefore, if we assume compression, there is only the possibility to first download the compressed backup and then decompress it in the target server. However, if we consider the possibility to not compress the backup, then the following equation shows the optimal ship time depending on the available bandwidth.

$$t_{ship} = \begin{cases} \frac{CS}{BW} + \frac{TS}{DS}, BW \leq \left(1 - \frac{1}{CR}\right) * DS \\ \frac{TS}{BW}, BW > \left(1 - \frac{1}{CR}\right) * DS \end{cases} \quad (2)$$

In the first interval, the bandwidth is small enough so that it is worthier to first transfer the compressed files and then decompress them. On the other hand, in the second interval the bandwidth is large enough so that it is better not to compress the files and avoid the decompression time. Considering the results discussed in Section V of this paper, only for the scenarios with a very fast access to the Internet (more than 1.5 Gbps of bandwidth) it is more interesting not to use compression. However, the use of compression is always more efficient in terms of storage.

4 RUN

The purpose of the run process is to take the generated files in the build process and obtain a running CiaB instance. In order to do so, the following steps are performed in sequence:

1) Bootstrap: it installs all the software needed to run CORD, e.g., Vagrant, Ansible, Docker, etc. This step is reused from the raw installation process of CORD.

2) Decompress CORD source tree: it decompresses the CORD source tree and saves the output in the user's home directory.

3) Prerequisites check: it runs an Ansible playbook retrieved in step 2 that checks that the machine where CORD is to be installed meets the hardware requirements (available RAM, CPU cores, etc.).

4) Decompress VM images: it decompresses the VMs and saves them in the corresponding path.

5) Start virtualization service: it makes sure that the virtualization service that manages the vagrant VMs is up and running.

6) Define VMs and networks: it defines the VMs and the virtual networks using the corresponding configuration files. Note that a default network already exists in the server before this step; therefore, it has to be removed before defining the default network configuration file generated during the "Build" process.

7) CORD configuration: it defines the configuration of the CiaB using the main makefile used in the raw deployment.

8) Build OvS: it installs and configures the required OvS switches.

9) Fire up VMs: it starts the Vagrant VMs.

These steps should be enough to get a fully-functionall CORD instance. However, there are some minor issues that still have to be fixed. These issues result in the following steps:

10) SSH configuration: a file that is copied from the CORD source tree is responsible for configuring the secure shell (SSH) connectivity to the VMs, which is specific to the user. Thus, if CORD is being installed in a different machine than the one the build process was performed on, the ssh connectivity will not work. In this step, this file is modified to make sure that it refers to the user in the deployment machine.

11) Restart Apache: sometimes a component in the head node called Keystone boots in a unstable state. The way to fix it is to restart its Apache server.

12) Fix iptables: sometimes, the connectivity between the head and the compute nodes is lost. This step fix this issue through an iptables command.

5 RESULTS

5.1 Compression Algorithms

Performance results of the proposed BSR approach for deploying CORD compared to the raw approach are analyzed in this section. The BSR approach has been tested using different compression algorithms. Traditional compression algorithms such as *XZ* (Colin, 2018), *BZIP2* (Seward, 1997) or *GZIP* (Deutsch, 1996) try to achieve higher compression rates at the cost of a lower decompression speed. However, this work required a compression algorithm with high decompression speeds that could keep a similar compression rate as the traditional algorithms since this would allow for fast deployment. Compression times are not a strict limitation as they are expected to influence only the initial "packing" of the CiaB deployment. An algorithm called *LZ4* (Collet, 2013) meets these requirements. Table 1 shows a comparison in terms of compression rate and decompression speed

among the aforementioned algorithms based on the compression of the CORD source tree.

Table 1: Compression algorithms comparison

Compression algorithm	Compression ratio	Decompression speed (MBps)
BZIP2	2,46	21,1
XZ	2,76	35,6
GZIP	2,33	79,1
LZ4-HC	2,01	371,8

It can be observed that the *LZ4* algorithms, configured in its high-compression mode (*LZ4-HC*) offers a decompression speed up to 5 times faster than *GZIP*, the runner-up in decompression speed, with a compression ratio a 27% worse than *XZ*, which has the best compression ratio.

5.2 Compression Algorithms

Choosing LZ4 as the best compression algorithm candidate, we performed different experiments with it. The total deployment time of CiaB is depicted in Fig. 1 for the different performed tests, as well as for a raw deployment. The "Ship" time is "omitted" for the BSR tests because it is assumed that the CORD backup is placed in a local repository and the transfer speed between this repository and the target server is larger than the decompression speed of the VMs (371.8 MBps), which reduces the shipping time to the decompression time. This is a reasonable assumption since, for example, SATA II (up to 3 Gbps (Serial ATA International Organization, 2007)) and USB 3.0 (up to 5 Gbps (Hewlett Packard, 2008)) support higher transfer speeds. If the backup was to be obtained from an online repository, the time to download it should be added into the presented results.

As a first approach, both the CORD source tree and the CORD VM images are first packed into a file with *tar* (Gilmore, 2008), and then these files are compressed with LZ4. The use of *tar* is necessary since LZ4 only supports the compression of files, and not folders or groups of files. However, since CORD consists in only five VM images, we also tested how our approach worked when compressing/decompressing the CORD images individually. These two approaches (called "LZ4 + tar" and "LZ4", respectively) are compared in Fig. 1, where it can be seen that the decompression time when not using *tar* for the VM images is clearly lower. A third approach was also tested based on the "LZ4" approach, i.e., without using *tar* for the images. In this third approach, some images (specifically, the "qcow2" format ones), are shrunk with a "qemu-img" (Bellard, 2014) command before being compressed. This reduces the disk space of the image, deleting actual information/data. As depicted in Fig. 1, this approach (called "LZ4 + shrink") also reduced the decompression time compared to the previous ones, whereas the VMs' firing up time kept constant. Moreover, it can be seen that the total deployment time of CiaB using the "LZ4 + shrink" BSR approach is around 7 times faster than using a raw installation, at about 12 minutes. The achieved deployment time, is well below the threshold of 90 minutes imposed by 5G-PPP (Kennedy, 2017)

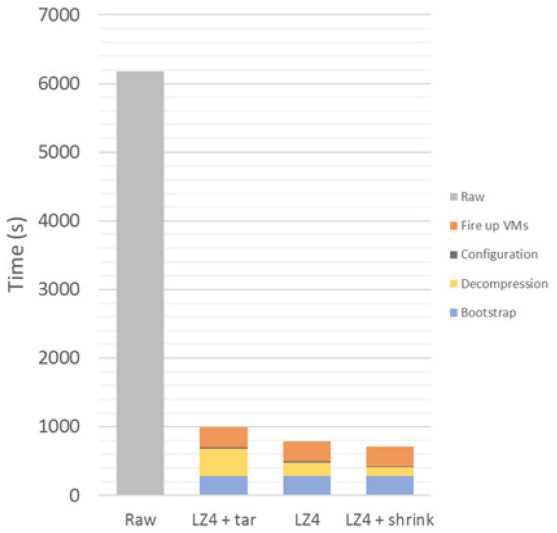

Figure 1 Results comparing a raw CiaB deployment time with the time obtained using different BSR approaches.

6 CONCLUSIONS

This paper investigated the use of compressed, pre-configured VM snapshots for the deployment of the CORD platform. The proposed deployment workflow follows a Build-Ship-Run approach and allows for much faster deployment times (7x) while also allowing for zero-touch configuration. Moreover, the BSR approach increases the reliability when deploying CiaB, meaning that the CORD code-base will not change from deployment to deployment, as it might happen among different raw deployments. Finally, the presented approach is also useful to save the state of a running CiaB deployment, i.e., to create a backup of the entire

system. Then this backup can be restored at any time in the same machine or a remote one. In addition to the proposed workflow, this paper also investigated the effects of different compression algorithms, with respect to the final size of the archives and their decompression times. The presented results show that “trimming” the VM images and then using the LZ4 compression provides with the best results. As future work, the presented approach will be extended to support newer CORD versions, as the focus has been given to CORD 4.1 so far.

ACKNOWLEDGEMENTS

This work has been performed in the framework of the NGPaaS project, funded by the European Commission under the Horizon 2020 and 5G-PPP Phase2 programmes, under Grant Agreement No. 761 557 (http://ngpaas.eu).

REFERENCES

Peterson, L., et al. 2016. "Central office re-architected as a data center". IEEE Communications Magazine 54.10 96-101.

InstaCORD, 2017. Available at: https://wiki.opencord.org/display/CORD/InstaCORD+on+CloudLab .

Collin, L., 2018. "XZ utils". Available at: http://tukaani.org/xz/

Seward, J., 1997. "The bzip2 home page". Available at: http://www.bzip.org

Deutsch, P., 1996. "GZIP file format specification version 4.3. No. RFC 1952".

Collet, Y., 2013. "LZ4-Extremely fast compression".

Serial ATA International Organization , 2007. Serial, A. T. A. "revision 2.6".

Hewlett-Packard Company, Intel Corporation, Microsoft Corporation, NEC Corporation, ST-NXP Wireless, Texas Intruments, 2008, Bus, Universal Serial. "3.0 Specification", rev 1

Gilmore, J., 2008. "Tar-GNU Project—Free Software Foundation (FSF)." Available at: http://www.gnu.org/software/tar

Bellard F., 2014. Ubuntu, “trusty (1) qemu-img.1.gz”. Available at: http://manpages.ubuntu.com/manpages/trusty/en/man1/qemu-img.1.html

Kennedy D., 2017. “Euro-5g –Supporting the European 5G Initiative, D2.6 Finalreport on programme progress and KPIs”