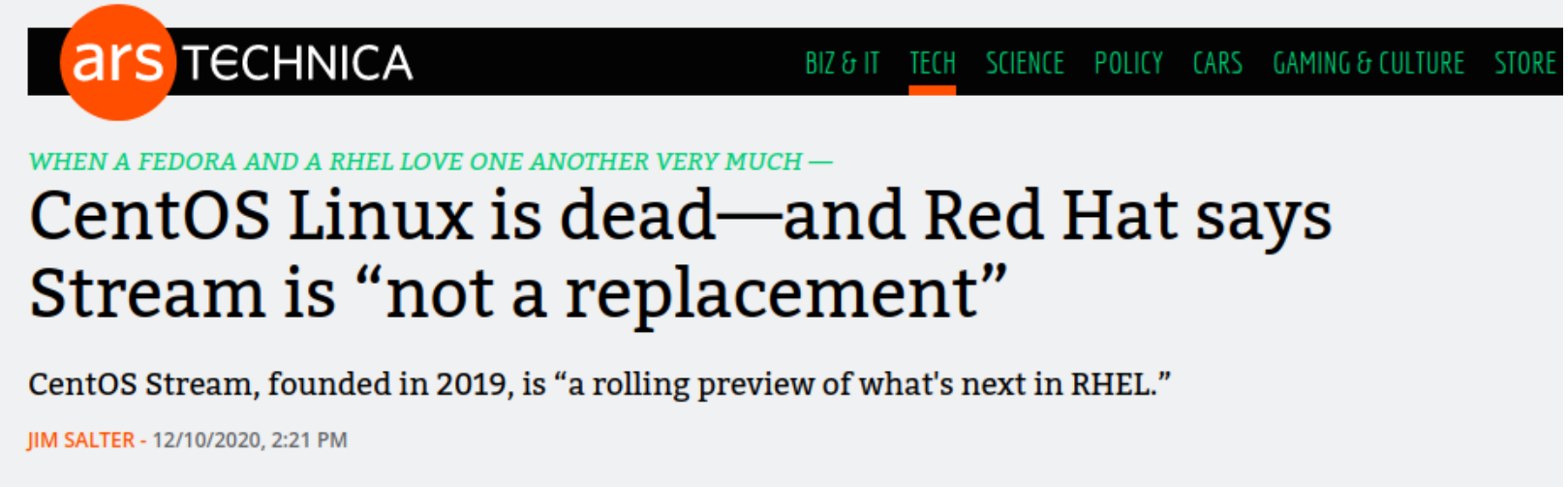
CERN
Tape Archive

Experiences Migrating CTA to Alma9

Jorge Camarero Vera

CERN IT-SD-TAB section

Why Alma9?



ars TECHNICA

BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE STORE

WHEN A FEDORA AND A RHEL LOVE ONE ANOTHER VERY MUCH —

CentOS Linux is dead—and Red Hat says Stream is "not a replacement"

CentOS Stream, founded in 2019, is "a rolling preview of what's next in RHEL."

JIM SALTER - 12/10/2020, 2:21 PM

https://arstechnica.com/gadgets/2020/12/centos-shifts-from-red-hat-unbranded-to-red-hat-beta/

CentOS Linux is reaching its end of life. Now what?

Updates and releases of CentOS Linux® 8 were discontinued on December 31, 2021, and will be discontinued for CentOS Linux 7 on June 30, 2024. If your organization is running part of your environment on CentOS Linux, attend this session to learn which use cases are best suited for CentOS Stream and which are more appropriate for Red Hat® Enterprise Linux. You will also learn about new programs that make Red Hat Enterprise Linux more accessible, streamline migrations, and require minimal staff retraining.

https://www.redhat.com/en/events/webinar/centos-linux-reaching-its-end-life-now-what

Why not Rocky8/9?

- CERN only supports both **RHEL** and **AlmaLinux** for the 8 or 9 families.
- Reasons why Alma9 was selected:

1. Community involvement
2. Package version policy: mimics RHEL
3. Transparency
4. Major/minor point release delay
5. Compatibility with CentOS SIGs
6. ...

https://linux.web.cern.ch/which/

Update on the Linux Strategy for CERN (and WLCG)

Migration to Alma9

The migration of the CTA codebase from **CentOS 7 to Alma9** presents a range of compatibility issues:

- **Dependency Management:** Navigating version changes in vital dependencies like Protobuf and Oracle Instant Client.
- **Distribution Discrepancies:** Resolving differences between CC7 and Alma9 distribution offerings.
- **Versionlock Adaptation:** Updating and relocating the versionlock file for Alma9.
- **Code Adaptation:** Ensuring code compatibility with potential C++ standard variations. Script modernization from Python 2 to Python 3.
- **Continuous Integration:** Integrating the CTA docker image into a minikube environment. Problems found with scripts and transitioning from Docker to Podman

Dependency Management

Library	Centos7 Version	Alma9 Version
Oracle Instant Client	19.3	21.12
Protobuf	3.3.1 (cern)	3.14.0 (System)
Gtest	1.12.0 (cern)	1.11.0 (System)
XRootD	4.12.8-1 / 5.6.1-1	5.6.1-1
Grpc	1.19.0	1.46.7 (System)
librados-devel	15.2.15	17.2.7 (System)
sqlite-devel	3.7.17	3.34.1 (System)
libcap-devel	2.22	2.48 (System)
binutils-devel	2.27	2.35.2 (System)
cryptopp-devel	5.6.2	8.6.0 (System)
libuuid-devel	2.23.2	2.37.4 (System)
json-c-devel	0.11	0.14 (System)
libattr-devel	2.4.46	2.5.1 (System)
postgresql-devel	9.2.24	13.11-1 (libpq-devel / System)
Valgrind	3.15.0	3.21.0-7 (System)
krb5-devel	1.15.1-55	1.21.1-1 (System)
libtirpc-devel	0.2.4	1.3.3 (System)

- Use of dependencies provided by Alma9.
- Major problems with Protobuf and Oracle Instant Client.
- Change of code behaviour in json-c-devel library.
- Downgrade Google Tests to use the provided one by Alma9

Dependency Management

Protobuf Alma9

```
set(PROTOBUF3_RPATH ${PROTOBUF3_ROOT}/lib64)
message(STATUS "PROTOBUF3_RPATH=${PROTOBUF3_RPATH}")

set(PROTOBUF3_INCLUDE_PATH ${CMAKE_CURRENT_SOURCE_DIR})

find_program(PROTOBUF3_PROTOC3_EXECUTABLE
    NAMES ${PROTOBUF3_ROOT}/bin/protoc
    DOC "Version 3 of The Google Protocol Buffers Compiler"
)
message(STATUS "protoc is at ${PROTOBUF3_PROTOC3_EXECUTABLE} ")

find_path(PROTOBUF3_INCLUDE_DIRS
  google/protobuf/message.h
  PATHS ${PROTOBUF3_ROOT}/include
  NO_DEFAULT_PATH)
message(STATUS "PROTOBUF3_INCLUDE_DIRS=${PROTOBUF3_INCLUDE_DIRS}")
```

Protobuf Centos7

```
set(PROTOBUF3_RPATH ${PROTOBUF3_ROOT}/lib64/protobuf3)
message(STATUS "PROTOBUF3_RPATH=${PROTOBUF3_RPATH}")

set(PROTOBUF3_INCLUDE_PATH ${CMAKE_CURRENT_SOURCE_DIR})

find_program(PROTOBUF3_PROTOC3_EXECUTABLE
    NAMES ${PROTOBUF3_ROOT}/bin/protoc3
    DOC "Version 3 of The Google Protocol Buffers Compiler"
)
message(STATUS "protoc is at ${PROTOBUF3_PROTOC3_EXECUTABLE} ")

find_path(PROTOBUF3_INCLUDE_DIRS
  google/protobuf/message.h
  PATHS ${PROTOBUF3_ROOT}/include/protobuf3
  NO_DEFAULT_PATH)
message(STATUS "PROTOBUF3_INCLUDE_DIRS=${PROTOBUF3_INCLUDE_DIRS}")
```

- Alma9 doesn't support **Protobuf 2**.
- **Protobuf 3** is by default in Alma9, so it doesn't need specific path as in Centos7

Dependency Management

Oracle Instant Client

```
if (ALMA9)
  find_library(ORACLE-INSTANTCLIENT_OCCI_LIBRARY
    NAME libocci_gcc53.so
    PATHS ${ORACLE-INSTANTCLIENT_RPATH}
    NO_DEFAULT_PATH)
else()
  find_library(ORACLE-INSTANTCLIENT_OCCI_LIBRARY
    NAME libocci.so.19.1
    PATHS ${ORACLE-INSTANTCLIENT_RPATH}
    NO_DEFAULT_PATH)
endif()
```

- Oracle Instant Client usually is compiled with ABI=0, we have to find packages with **libocci_gcc53.so** to work with Alma 9.
- An **installation script**: continuousintegration/docker/ctafrontend/alma9/installOracle21.sh

Distribution Discrepancies

- Some packages are installed by default in Centos7 and no in Alma9: **expect** package is necessary to use the process **unbuffer**.
- Some library headers of library are distributed by different packages: **rpc** headers are now provided by **libtirpc-devel**.
- Main system programs can has different paths: In Alma9 **nologin** is in /usr/sbin/nologin and in Centos7 in /bin/nologin

Versionlock Adaptation

Centos7

```
0:eos-archive-5.2.8-1.el7.cern.x86_64.rpm
0:eos-cleanup-5.2.8-1.el7.cern.x86_64.rpm
0:eos-client-5.2.8-1.el7.cern.x86_64.rpm
0:eos-fuse-5.2.8-1.el7.cern.x86_64.rpm
0:eos-fuse-core-5.2.8-1.el7.cern.x86_64.rpm
0:eos-fuse-sysv-5.2.8-1.el7.cern.x86_64.rpm
0:eos-fusex-5.2.8-1.el7.cern.x86_64.rpm
0:eos-fusex-core-5.2.8-1.el7.cern.x86_64.rpm
0:eos-fusex-selinux-5.2.8-1.el7.cern.x86_64.rpm
0:eos-ns-inspect-5.2.8-1.el7.cern.x86_64.rpm
0:eos-server-5.2.8-1.el7.cern.x86_64.rpm
0:eos-srm-5.2.8-1.el7.cern.x86_64.rpm
0:eos-test-5.2.8-1.el7.cern.x86_64.rpm
0:eos-testkeytab-5.2.8-1.el7.cern.x86_64.rpm
0:eos-quarkdb-5.2.8-1.el7.cern.x86_64.rpm
```

Alma9

```
eos-archive-5.2.8-1.el9.x86_64
eos-cleanup-5.2.8-1.el9.x86_64
eos-client-5.2.8-1.el9.x86_64
eos-fuse-5.2.8-1.el9.x86_64
eos-fuse-core-5.2.8-1.el9.x86_64
eos-fuse-sysv-5.2.8-1.el9.x86_64
eos-fusex-5.2.8-1.el9.x86_64
eos-fusex-core-5.2.8-1.el9.x86_64
eos-fusex-selinux-5.2.8-1.el9.x86_64
eos-ns-inspect-5.2.8-1.el9.x86_64
eos-server-5.2.8-1.el9.x86_64
eos-srm-5.2.8-1.el9.x86_64
eos-test-5.2.8-1.el9.x86_64
eos-testkeytab-5.2.8-1.el9.x86_64
eos-quarkdb-5.2.8-1.el9.x86_64
```

- In Alma9 version lock format is more **simple**
- In Alma9 the entries of the versionlock file doesn't include the extension for the **rpm** (.rpm)
- Also in Alma9 the versionlock is located in: /etc/dnf/plugins/versionlock.list
- But in Centos7 is located in: /etc/yum/pluginconf.d/versionlock.list

Code Adaptation: C++

New method in Json

```
template<>
void JSONCObject::jsonSetValue(const std::string& key, const double & value){
#ifdef ALMA9
  char buffer[64];
  snprintf(buffer, sizeof(buffer), "%.6f", value);
  json_object_object_add(m_jsonObject,key.c_str(),json_object_new_double_s(value, buffer));
#else
  json_object_object_add(m_jsonObject,key.c_str(),json_object_new_double(value));
#endif
}
```

- The output for '42.0' in Centos7 is '42.000000', but in Alma9 is '42.0'. So to achieve same value in Alma9 we will use the method **json_object_new_double_s**.

SubProcess

```
TEST(SubProcessHelper, basicTests) {
  cta::threading::SubProcess sp("/usr/bin/echo", std::list<std::string>({"/usr/bin/echo", "Hello,", "world."}));
  sp.wait();
  ASSERT_EQ("Hello, world.\n", sp.stdout());
  ASSERT_EQ("", sp.stderr());
  ASSERT_EQ(0, sp.exitValue());
  cta::threading::SubProcess sp2("/usr/bin/cat", std::list<std::string>({"/usr/bin/cat", "/no/such/file"}));
  sp2.wait();
  ASSERT_EQ("", sp2.stdout());
  ASSERT_NE(std::string::npos, sp2.stderr().find("/no/such/file"));
  ASSERT_EQ(1, sp2.exitValue());
  #ifdef ALMA9
    EXPECT_THROW(cta::threading::SubProcess sp3("/no/such/file", std::list<std::string>({"/no/such/file"})),
      cta::exception::Errnum);
  #else
    cta::threading::SubProcess sp3("/no/such/file", std::list<std::string>({"/no/such/file"}));
    sp3.wait();
    ASSERT_EQ("", sp3.stdout());
    ASSERT_EQ(127, sp3.exitValue());
    ASSERT_EQ("", sp3.stderr());
  #endif
}
```

- If the process to be called doesn't exists **Alma9** produce an **exception**, but in **Centos7** no.

- Also Alma9 doesn't find the process if the absolute path is not passed. In **Centos7** is enough with "echo" or "cat", but in **Alma9** is"/usr/bin/echo" or "/usr/bin/cat"

Continuous Integration: Containerization Shift

- In Alma9 we will use **podman** instead of **docker** for the containerization for the kubernetes tests. And for kubernetes we will use **Minikube**.
- This shift is motivated by kubernetes **deprecating docker** as container engine.
- Installation of minikube for CTA
- For Minikube in Alma9 we have to **load the image manually** in the kubernetes cluster:

```
# Step 1: Save your Podman image to a tar file
podman save localhost/ctageneric:dev -o ctageneric.tar

# Step 2: Load your Podman image into Minikube
minikube image load ctageneric.tar

# Step 3: Verify your image is in Minikube
minikube image ls
```

Code Adaptation: Python Migration

- Alma9 doesn't support python2 (it reached end of life 1st January 2020). So we had to migrate to python3 the scripts: **ctafstgcd.py** and **test_ctafstgcd.py**

- The python script **cta-versionlock** hasn't been migrated to python3 yet. But for the current development is not necessary.

Continuous Integration: Shell Scripting

- Different bash version. In Centos7 it's **4.2.46** and in Alma9 it's **5.1.8**. This lead to different behaviours.

- In Alma9 to get the same output as `**IFS=' ' read -r -a**` we have to use `**readarray -t**`

```
# CC7 doesn't have readarray
if [ "$(cat /etc/redhat-release | grep -c 'AlmaLinux release 9')" -eq 0 ]; then
  echo "This container is not running on AlmaLinux 9"
  IFS=' ' read -r -a dr_names <<< $(admin_cta --json dr ls | jq -r '.[] | select(.driveStatus=="UP") | .driveName')
  IFS=' ' read -r -a dr_names_down <<< $(admin_cta --json dr ls | jq -r '.[] | select(.driveStatus=="DOWN") | .driveName')
  IFS=' ' read -r -a vids <<< $(admin_cta --json ta ls --all | jq -r '.[] | .vid')
  IFS=' ' read -r -a lls <<< $(admin_cta --json ll ls | jq -r '.[] | .name')
else
  echo "This container is running on AlmaLinux 9"
  readarray -t dr_names <<< $(admin_cta --json dr ls | jq -r '.[] | select(.driveStatus=="UP") | .driveName')
  readarray -t dr_names_down <<< $(admin_cta --json dr ls | jq -r '.[] | select(.driveStatus=="DOWN") | .driveName')
  readarray -t vids <<< $(admin_cta --json ta ls --all | jq -r '.[] | .vid')
  readarray -t lls <<< $(admin_cta --json ll ls | jq -r '.[] | .name')
fi
```

Continuous Integration

- https connection certificates for **client_rest_api** were failing in Alma9, so they were disabled.
- Podman image in Alma9 is in another path. It adds **localhost/** in the registry of minikube.

```
if [ ! -z "${dockerimage}" ]; then
  echo "set image to ctageneric:${imagetag}"
  if [ "$(cat /etc/redhat-release | grep -c 'AlmaLinux release 9')" -eq 0 ]; then
    echo "Not running on AlmaLinux 9"
    sed -i ${poddir}/pod-* -e "s/\(^\s\+image\):.*/\1: ctageneric:${imagetag}\n\1PullPolicy: Never/"
  else
    echo "Running on AlmaLinux 9"
    sed -i ${poddir}/pod-* -e "s/\(^\s\+image\):.*/\1: localhost\/ctageneric:${imagetag}\n\1PullPolicy: Never/"
  fi
else
  sed -i ${poddir}/pod-* -e "s/\(^\s\+image:[^:]\+:\).*/\1${imagetag}/"
fi
```

Conclusions and Lessons

1. Successfully transitioned CTA from CentOS 7 to Alma9.
2. Overcame challenges with key dependencies (e.g., Protobuf, Oracle Instant Client).
3. Ensured compatibility and functional integrity.
4. Resolved package and library header discrepancies.
5. Adapted to different system environments and paths.
6. Updated versionlock management and code standards for C++ and Python.
7. Transitioned from Docker to Podman, embracing new containerization practices.
8. Improved CI processes with Kubernetes and addressed scripting and HTTPS nuances.
9. Ensured more robust and efficient development pipelines.

home.cern