Asynchronous Network Management System (ANMS) Product Guide

DOC-005444, Prepared by The Johns Hopkins Applied Physics Laboratory

License

This work was performed for the Jet Propulsion Laboratory, California Institute of Technology, sponsored by the United States Government under the prime contract 80NM0018D0004 between the Caltech and NASA under subcontract 1658085.

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME
Initial	30 August 2023	Initial issue of document for ANMS v1.0	

Contents

Introduction viii

Identification . . . viii

Scope . . . viii

Terminology . . . viii

References . . . ix

1 ANMS Architecture 1

1.1 ANMS Components . . . 1

1.2 Deployment . . . 3

1.2.1 Using a CAM Gateway Emulator . . . 4

1.3 Containers . . . 4

1.4 Filesystem . . . 5

1.5 Networking . . . 5

1.6 Security . . . 6

1.7 Long-Term Databases . . . 6

2 Procedures 7

2.1 Building . . . 7

2.2 Installation . . . 8

2.2.1 TLS Configuration Volume . . . 8

2.3 Upgrading . . . 9

2.4 Resetting Docker State . . . 9

2.5 Monitoring . . . 9

2.6 Long-Term Database Backup and Restore . . . 9

2.6.1 Docker State and Logs . . . 9

2.6.2 SELinux Audit Events . . . 10

2.7 Checkout Procedures . . . 10

2.7.1 Frontend Communication . . . 10

2.7.2 BP Agent Communication . . . 11

3 Product Support **12**

3.1 Troubleshooting . . . 12

3.1.1 Installation . . . 12

3.1.1.1 SELinux Blocked Behavior . . . 12

3.1.2 Operations . . . 12

3.1.2.1 Grafana Containers . . . 12

3.1.2.2 Agent Registration Issues on ANMS Startup . . . 13

3.1.2.3 New Agent Registration Issues . . . 13

3.1.2.4 AMP Database Querying . . . 13

3.1.2.5 ANMS-UI is not visible at hostname:9030 . . . 13

3.1.2.6 ANMS-UI is not visible at hostname . . . 13

3.2 Contacting or Contributing . . . 14

Index **15**

List of Figures

1.1 ANMS Components with Protocol Associations 2
1.2 ANMS Primary Components with Logical Associations 3

List of Tables

1 Applicable JPL Rules Documents ix
2 Applicable MGSS Documents ix
3 Applicable Other Documents ix

1.1 Target host packages 4

Introduction

This Product Guide provides architectural and maintenance details about the Asynchronous Network Management System (ANMS), which is part of the Advanced Multi Mission Operations System (AMMOS) suite of tools.

Identification

Property	Value
Configuration ID (CI)	631.17
Element	Multi-Mission Control System (MMCS)
Program Set	Asynchronous Network Management System (ANMS)
Version	1.0

Scope

This document describes technical details about the ANMS installation, upgrade, monitoring, and maintenance. For details about the user interface and workflows of the ANMS see the ANMS User Guide.

Terminology

Asynchronous Management Protocol (AMP)

The application protocol used to communicate between ANMS and its managed agents.

Bundle Protocol (BP)

The overlay network protocol used to transport AMP messages between ANMS and its managed agents.

BP Agent (BPA)

The instantiation of a BP node with a unique administrative Endpoint ID.

BP Endpoint

The source or destination of a BP bundle.

Container

An isolated unit of runtime state within a host.

Convergence Layer Adapter (CLA)

The mechanims used to transport bundles within an underlay (IP) network.

Endpoint ID (EID)

The identifier of a BP Endpoint; names the source and destination for a BP bundle.

Host

A single node on the network and a single instance of an operating system. One host can have many interfaces and many IP addresses, but only one canonical host name.

Internet Protocol (IP)

The network protocol used for inter-container communication and for BP convergence layer messaging with AMP Agents.

References

Title	Document Number
Software Development	57653 rev 10

Table 1: Applicable JPL Rules Documents

Title	Document Number
MGSS Implementation and Maintenance Task Requirements	DOC-001819 ref F
Common Access Manager (CAM) Product Guide (PG)	DOC-005065
ANMS Architecture Description Document	DOC-005089
ANMS Software Design Document	DOC-005445
ANMS Software Interface Specification	DOC-005446
ANMS User Guide	DOC-005443

Table 2: Applicable MGSS Documents

Title	Reference
Installing Puppet	puppet-agent
Installing Bolt	puppet-bolt
Using SELinux	rhel8-selinux
ANMS Source	anms-source
ANMS Guide Document Source	anms-docs

Table 3: Applicable Other Documents

Chapter 1

ANMS Architecture

The ANMS is designed with a microservice architecture using containers for isolation and network protocols to communicate between services. Many of the service interfaces use stateless HTTP exchanges, some use MQTT for pub-sub interactions and some use PostgreSQL for long-term configuration and data warehousing. Although some of the subsystems have a preferred start-up order (to ensure initial configuration is valid and consistent) the containers can be restarted in any order with a few specific exceptions.

1.1 ANMS Components

The subsystems of the ANMS (all Docker containers) are illustrated as gray blocks within the "ANMS Instance" group in the diagram of Figure 1.1. The entire ANMS instance is made to be run on a single host, with future plans to allow installing in a more distributed environment. Currently the ANMS provides security at the boundary of the instance but not between comtainers (see Section 1.6 for details), which would be required for a distributed installation.

A higher-level logical view of the ANMS is shown in Figure 1.2 where some of the internal infrastructure containers (e.g., PostgreSQL database, MQTT broker) are removed for clarity.

The User Agents in both diagrams are how a user can interact with the ANMS, which is solely via HTTP exchanges. Most of the ANMS API follows RESTful principals of stateless service interactions, while some of the API is more browser-oriented to provide UI visuals.

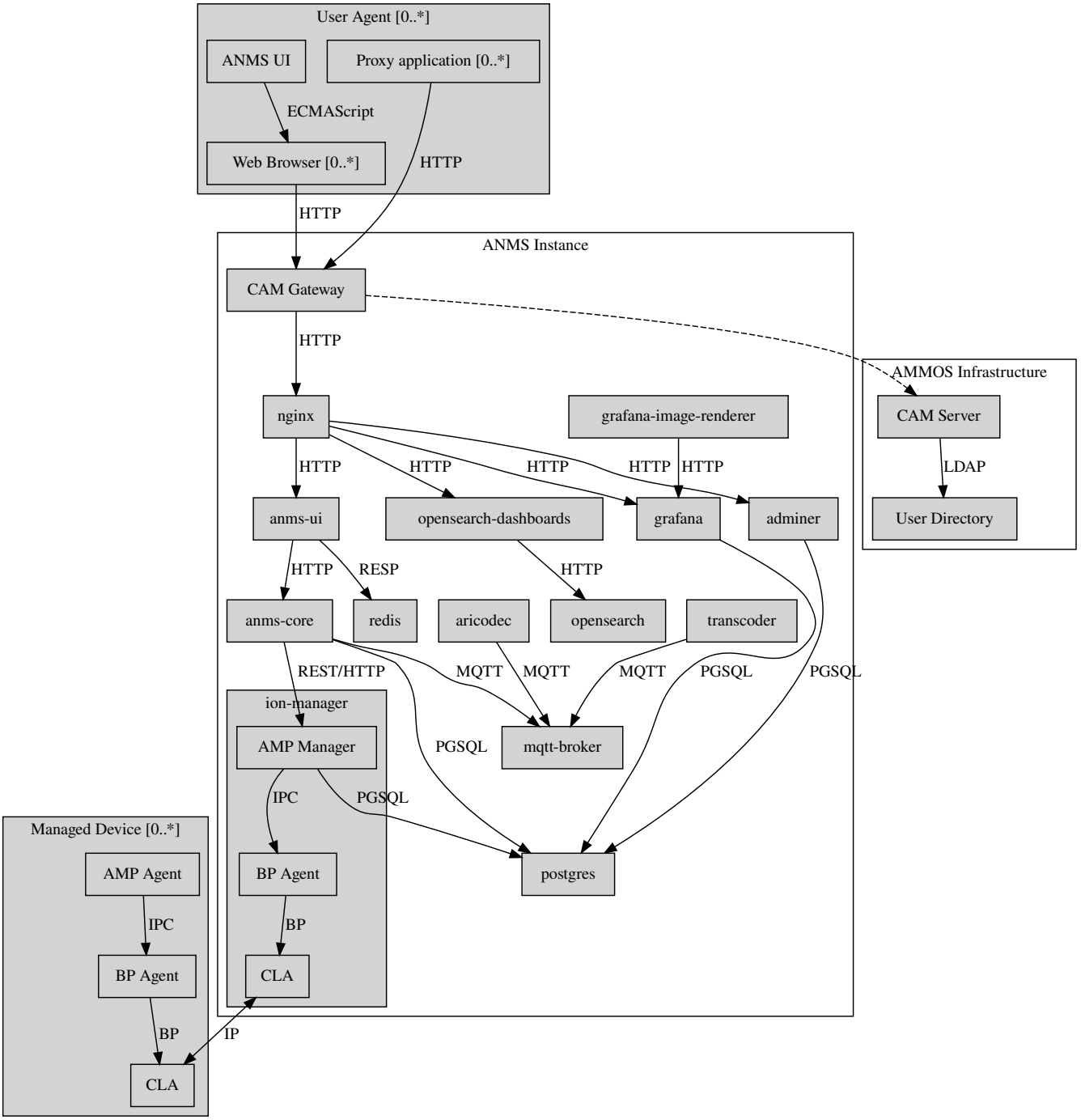

Figure 1.1: ANMS Components with Protocol Associations

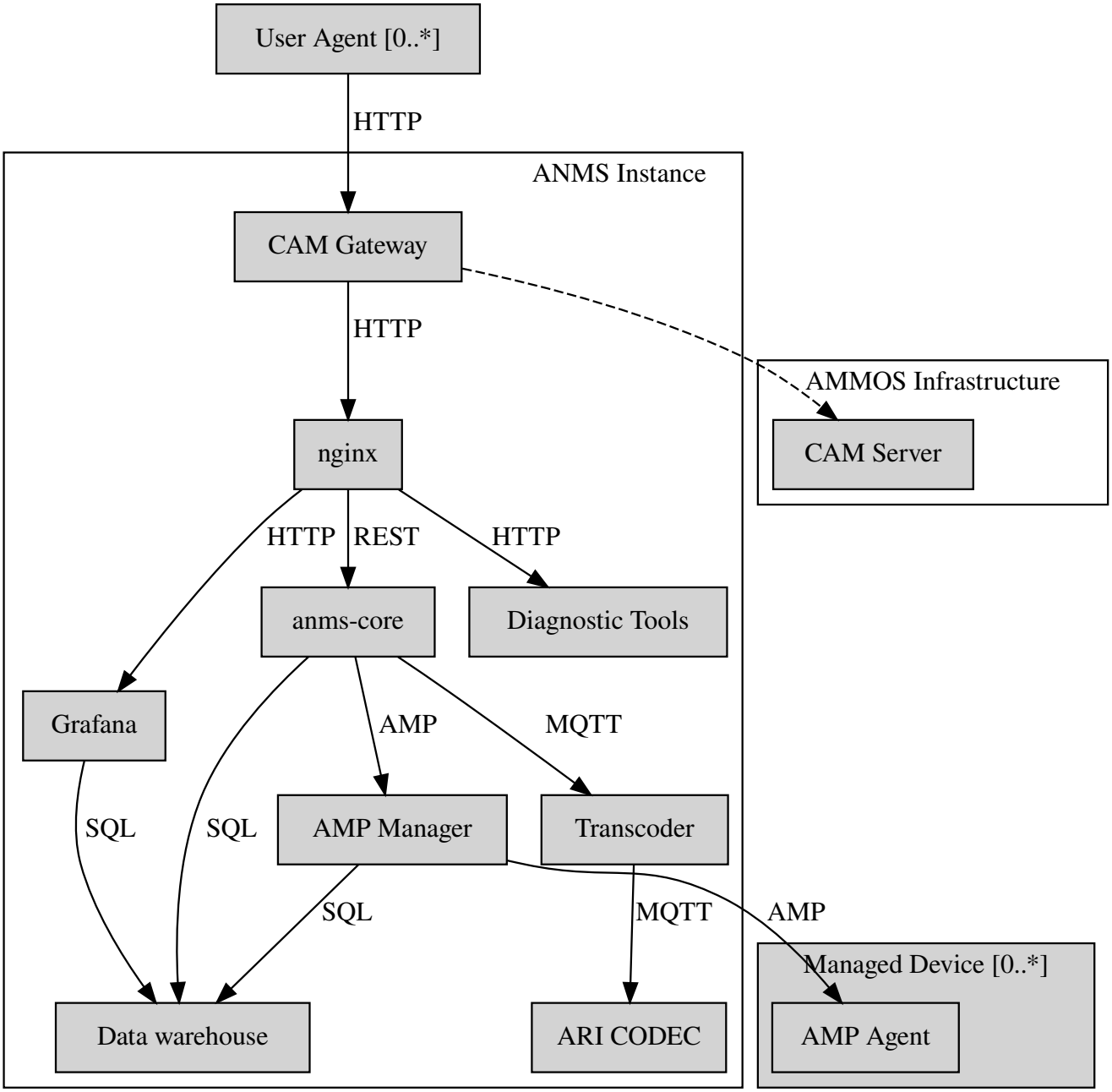

Figure 1.2: ANMS Primary Components with Logical Associations

1.2 Deployment

The target host will be running the RedHat Enterprise Linux (RHEL) version 8 (RHEL-8) with network interfaces configured, and IP addressing and DNS configured along with a running local firewall.

The ANMS is intended to be deployed using the Puppet orchestrator, either from a local Puppet apply execution or configured from a central Puppet server. Part of the ANMS source is a Puppet module "anms" to automate the configuration of an ANMS deployment. Specific procedures for performing an installation using a local Puppet apply are in Section 2.2.

Conditions for installing the ANMS are a host with packages identified in Table 1.1, at least 7 GiB of filesystem space for docker image storage, and additional space for long-term data warehouse storage. The total amount of storage needed depends on the

mission use of reports, specifically the average size and rate of reported data.

Package Name	Version Minimum
docker-ce	23.0.1
docker-compose	1.29.2
Puppet	7

Table 1.1: Target host packages

The ANMS is designed to operate on a network where the MGSS Common Access Manager (CAM) is used to manage user accounts and a CAM Gateway is used as a reverse proxy within the ANMS installation to enforce user login sessions and access permissions. The ANMS has been exercised with CAM v5.1.0 in a test environment outside of the MGSS environment. To deploy the ANMS in an environment without a CAM instance available (or without using it) the ANMS can be built with a CAM Gateway emulator as described in Section 1.2.1. In any case, deployment and configuration of CAM itself is outside the scope of this document and is described in detail in the CAM Product Guide.

1.2.1 Using a CAM Gateway Emulator

To allow the ANMS to be tested in environments where a CAM Server is unavailable or too burdensome to set up, the ANMS can be built with an emulator of the CAM Gateway which uses static accounts, credentials, and access controls. The environment `AUTHNZ_EMU=1` during a build (see Section 2.1) enables the CAM Gateway emulator behavior.

Caution
The CAM Gateway emulator is for demonstration only and must not be present in a production installation.

The static accounts available in the emulator, defined in an `htpasswd` file, are:

`test` With password `test`, is able to access all typical ANMS UI and features.

`admin` With password `admin`, is able to access all typical ANMS UI and features as well as the `/adminer/...` and `/nm/...` APIs.

1.3 Containers

The containers defined by the ANMS compose configuration in Section 1.4 are as follows in alphabetical order. Associations between these containers are illustrated in Figure 1.1.

`adminer` Administrative access to the PostgreSQL database, which requires special authorization. Exposes TCP port 8080 for HTTP.

`authnz` The CAM Gateway reverse proxy for authentication, authorization, and auditing (AAA); also the endpoint of user agent TLS connections. This container uses the external `ammos-tls` volume for TLS configuration (see Section 2.2.1). Exposes TCP port 443 for HTTPS and 80 for HTTP, both mapped to the same host port numbers.

`anms-core` The ANMS backend REST services. Exposes TCP port 5555 for HTTP.

`anms-ui` The ANMS frontend REST services and browser client UI. Exposes TCP port 9030 for HTTP.

`grafana` The data warehouse plotting engine. This uses the `grafana-data` volume for storage. Exposes TCP port 3000 for HTTP.

grafana-image-renderer Image renderer for the `grafana` subsystem. Exposes TCP port 8081 for internal APIs.

ion-manager A combination of the AMP Manager used by ANMS and the BP Agent used for message transport. Exposes UDP port 1113 for LTPCL and port 4556 for UDPCL, and TCP port 8089 for HTTP API; the CL ports are mapped to the same host port numbers.

aricodec A service to convert ARI between text and compressed binary form based on available ADMs.

mqtt-broker The broker host for MQTT pub-sub messaging. Exposes TCP port 1883 for MQTT.

nginx Frontend load balancer and HTTP router. Principal access is to `anms-ui` and `grafana`. Exposes TCP port 80 for HTTP.

opensearch Log aggregator for the ANMS. This uses the `opensearch` volume for storage. Exposes TCP port 9200 and 9600 for internal APIs.

opensearch-dashboards A user interface for accessing the opensearch logs. Exposes TCP port 5601 for HTTP.

postgres Persistent database for the ANMS. This uses the `postgres-data` volume for storage. Exposes TCP port 5432 for PSQL.

redis A database for state keeping from the ANMS UI. Exposes TCP port 6379 for redis API.

transcoder An intermediate service to bookkeep transcoding requests from the ANMS to the ARI CODEC engine.

1.4 Filesystem

Because the ANMS is deployed as a Docker Compose configuration, the only primary files present on the host are to configure docker, its use as a system service, and the system firewall.

The principal files and directories used by ANMS are:

/ammos/anms The project-specific deployment path for compose configurations, under which are:

- **.env** Environment configuration for the ANMS containers.
- **docker-compose.yml** The actual Docker Compose configuration for the ANMS, which is configured for auto-startup containers at Docker start.

Secondary files related to the ANMS deployment are:

/etc/docker/daemon.json Configured to enable SELinux for containers.

/var/cache/puppet/puppet-selinux/modules The containing directory for SELinux modules for the ANMS containers (see Section 1.6).

1.5 Networking

The target host will be running RHEL-8 with network interfaces configured, and IP addressing and DNS configured along with a running local firewall.

The Docker network configuration for the ANMS includes host port forwarding for the following services:

HTTPS Default port 443 forwarded to the `authnz` container for HTTP use.

UDPCL Default port 4556 forwarded to the `ion-manager` container for BP use.

LTPCL Default port 1113 forwarded to the `ion-manager` container for BP use.

The current ANMS will allow only the BP UDP Convergence Layer (UDPCL) to be configured on agents, but this is a UI restriction and not an intrinsic limitation of the BP Agent used by the ANMS. Future versions of the ANMS will allow more complex convergence layer configurations.

Note
The ANMS deployment manifest includes an optional set of local AMP Agents to use to test with. These use the local Docker network to communicate with the ANMS, while real remote agents will require the external network configuration to include port forwarding and host name resolution for the ANMS BP Agent. How those are configured is outside the scope of this document.

1.6 Security

The host on which the ANMS instance runs is expected to have FIPS-140 mode enabled and SELinux enabled and in enforcing mode. Part of the ANMS deployment includes an SELinux module for each of the component containers which allow all necessary inter-service communication. If issues with SELinux are suspected in a deployment, follow the procedures in Section 2.6.2 to find any audit events related to the ANMS.

The host is also expected to have a running OS-default firewall which will be configured by the Puppet module to allow HTTPS for user agents and BP UDPCL and LTPCL default ports.

The interface between ANMS and its User Agents is TLS-secured HTTP with a PKIX certificate supplied by the host network management and chained to the CA hierarchy of the network.

The interface between ANMS and its managed AMP Agents is not currently secured, pending updates to the BP Agent to enable BPSec for integrity and/or confidentiality of AMP messages.

1.7 Long-Term Databases

The ANMS uses an internal PostgreSQL database for following purposes, all within the same schema `amp_core`:

User Configuration Each user account authorized to access the ANMS can have parameters associated with their account, mostly related to UI parameters.

ADM configuration The most static configuration of the ANMS is the set of ADMs available to all agents managed by that ANMS instance.

Agent configuration The Agent configuration consists of agents which are known to, and managed by, the ANMS which are parameterized by their AMP messaging BP EID and their associated CL parameters (network name/address and port).

Reported Data Warehouse When reports arrive from managed agents and are associated with known ADMs they are disassembled and stored as object-values in the historical data warehouse.

Chapter 2

Procedures

This chapter includes specific procedures related to managing an ANMS instance.

2.1 Building

The ANMS source is composed of a top-level repository `ammos-anms` and a number of submodule repositories; all of them are required for building the ANMS.

1. The top-level checkout can be done with:

   ```
   git clone --recursive --branch <TAGNAME> <BASEURL>/ammos-anms.git
   ```

2. Optional: switching to a different tag or branch can be done with the sequence:

   ```
   git checkout <TAGNAME>
   git submodule update --init --recursive
   ```

3. If necessary, add the local user to the `docker` access group with:

   ```
   sudo usermod -a -G docker ${USER}
   ```

4. The container image building is then executed with:

   ```
   export DOCKER_IMAGE_PREFIX=<DOCKERURL>
   export DOCKER_IMAGE_TAG=latest
   ./build.sh buildonly
   ```

   which by default uses the current top-level branch name as the tag for all container images.

Note
To build an ANMS that uses an emulator for the CAM Gateway (which means that the ANMS will not require a CAM server), have the following environment set in the build step above:

```
export AUTHNZ_EMU=1
```

2.2 Installation

The ANMS uses Puppet version 7 [puppet-agent] to install requisite system packages and configure system files and services. In addition, Bolt [puppet-bolt] is needed to install needed Puppet modules and run the Puppet agent remotely.

Caution
The example TLS configuration in this proceure is for demonstration only and must not be present in a production installation. Details for creating a proper TLS volume are in Section 2.2.1.

To install the ANMS on the local host perform the following:

1. A TLS configuration must be embedded in a volume mounted by the `authnz` frontend container with contents described in Section 2.2.1. This can be done with a boilerplate test-only CA and certificates by running:

   ```
   ./create_volume.sh ./puppet/modules/apl_test/files/anms/tls
   ```

2. The deployment configuration is set by editing the file `puppet/data/override.yaml` to contain similar to:

   ```
   anms::docker_image_prefix: "" # Matching the DOCKER_IMAGE_PREFIX from build procedure
   anms::docker_image_tag: "latest" # Matching the tag name from build procedure
   anms::docker_registry_user: ""
   anms::docker_registry_pass: ""
   ```

3. Pull the necessary upstream Puppet modules with:

   ```
   ./puppet/prep.sh
   ```

4. Perform a dry-run of the puppet apply with:

   ```
   sudo PATH=/opt/puppetlabs/bin:$PATH ./puppet/apply_local.sh --test --noop
   ```

   and verify that there are no unexpected changes.

5. Perform the actual puppet apply with:

   ```
   sudo PATH=/opt/puppetlabs/bin:$PATH ./puppet/apply_local.sh --test
   ```

2.2.1 TLS Configuration Volume

The docker volume mounted into the `authnz` container follows the AMMOS conventions for TLS certificate, private key, and CA file paths and contents.

The volume must contain the specific files:

`/certs/ammos-server-cert.pem` The PEM-encoded certificate for the ANMS frontend itself. It must have extended key use of `id-kp-serverAuth` for a web server.

`/private/ammos-server-key.pem` The PEM-encoded non-password-protected private key corresponding to the server certificate.

`/certs/ammos-ca-bundle.crt` The PEM-encoded CA bundle containing at least the CA chain used to sign the server certificate.

2.3 Upgrading

Because the ANMS is deployed as a Docker Compose configuration with associated environment variables and container images, an upgrade involves updating the compose configuration and restarting affected containers.

An upgrade can be performed using the same procedure as Section 2.2, where Puppet will make any required changes for the upgrade and restart services and containers as necessary. Individual ANMS releases may identify pre-upgrade or post-upgrade steps in their specific Release Description Document (RDD).

2.4 Resetting Docker State

Warning
The following will reset all database state, including user profiles, ADM configuration, and all historical report data. This should only be used for test hosts or after performing a full Postgres DB backup (see Section 2.6).

To force containers and volumes (containing long-term database files) to be cleared, a maintainer can run the following from the host.

```
docker stop $(docker ps -q); docker rm $(docker ps -a -q); docker volume rm $(docker volume ↩
    ls -q)
```

After clearing containers and volumes, the normal `apply_local` step of Section 2.2 should be performed to re-install and start the containers.

2.5 Monitoring

To enable Wireshark logging with patched AMP dissector, run similar to the following:

```
wireshark -i br-anms -f 'port 1113 or port 4556' -k
```

2.6 Long-Term Database Backup and Restore

Although the docker volume `anms_postgres` contains the raw database state, this will not allow backup of or transferring that state to other hosts.

To perform an online backup (keeping the database running) run the following on the host:

```
docker exec postgres pg_dump -Ft -d amp_core | gzip -c >~/anms-backup.tar.gz
```

which can the later be restored using:

```
gunzip -c <~/anms-backup.tar.gz | docker exec -i postgres pg_restore --clean -d amp_core
```

2.6.1 Docker State and Logs

Because of the Docker Compose configuration described in Section 1.4, accessing docker state and logs requires running docker with a command similar to the following:

```
docker-compose -f /ammos/anms/docker-compose.yml -p anms [action] ...
```

The state of all containers in the ANMS project can be observed with:

```
docker-compose -f /ammos/anms/docker-compose.yml -p anms ps
```

which will report a "State" column either as "Up" for simple servcies or "Up (healthy)" for health-checked services.

And observing logs from specific docker containers requires running a command similar to:

```
docker-compose -f /ammos/anms/docker-compose.yml -p anms logs [service-name]
```

2.6.2 SELinux Audit Events

The procedures in this section are a summary of more detail provided in Chapter 5 of the RedHat [rhel8-selinux] document.

By default, the `setroubleshootd` service is running, which intercepts SELinux audit events

To observe the system audit log in a formatted way run:

```
sudo sealert -l '*'
```

Some SELinux denials are marked as "don't audit" which suppresses normal audit logging when they occur. They are often associated with network access requests which would flood an audit log if they happen often and repeatedly. To enable logging of `dontaudit` events run:

```
sudo semanage dontaudit off
```

2.7 Checkout Procedures

Each of the following checkout procedures makes progressively more detailed and more normal-operations-like tests of the external interfaces with the ANMS.

In many fault cases, the procedure will work for the first steps and then fail on a specific step and thereafter. This is taken advantage of for the purposes of troubleshooting and failure reporting; the specific procedure(s) run and step(s) that fail are valuable to include in issue reports related to the ANMS.

To make the procedures more readable, the ANMS host is assumed to have the resolveable host name `anms-serv`. For checkout steps ocurring on a "client host" it is assumed to be running RHEL-8 or equivalent from the perspective of commands available.

2.7.1 Frontend Communication

This procedure checks the mechanism that a user agent can communicate with the ANMS just as a browser or user application would.

The checkout procedure is as follows:

1. From the ANMS host verify firewall access with:

   ```
   sudo firewall-cmd --zone public --list-services
   ```

   which should include the servies "https".

2. From a client host check the port is open with:

   ```
   nmap anms-serv -p80,443
   ```

   which should show a result similar to

```
PORT    STATE  SERVICE
80/tcp  closed http
443/tcp open   https
```

3. From a client host check HTTP access with:

   ```
   curl --head https://anms-serv/
   ```

   which should show a result containing lines similar to

   ```
   HTTP/1.1 302 Found
   Server: Apache/2.4.37 (Red Hat Enterprise Linux) OpenSSL/1.1.1k
   Location: /authn/login.html
   ```

4. From a client host check a test login account with:

   ```
   curl --head --user test https://anms-serv/
   ```

   along with the credentials for that account, which should show a result containing lines similar to

   ```
   HTTP/1.1 302 Found
   Server: Apache/2.4.37 (Red Hat Enterprise Linux) OpenSSL/1.1.1k
   Location: /authn/login.html
   ```

2.7.2 BP Agent Communication

This procedure checks whether the BPA in the ANMS can communicate with the BPA of a specific managed device.

The checkout procedure is as follows:

1. From the ANMS host verify firewall access with:

   ```
   sudo firewall-cmd --zone public --list-services
   ```

   which should include the servies "ltp" and "dtn-bundle-udp".

2. From any RHEL-8 host on the agent network run the following:

   ```
   sudo nmap anms-serv -sU -p4556
   ```

   which should show a result similar to

   ```
   PORT     STATE    SERVICE
   4556/udp filtered dtn-bundle-udp
   ```

3. From the ANMS host run the following, substituting the host name/address of any valid BP Agent:

   ```
   sudo nmap amp-agent -sU -p4556
   ```

   which should show a result similar to

   ```
   PORT     STATE    SERVICE
   4556/udp filtered dtn-bundle-udp
   ```

4. From the ANMS host run the following:

   ```
   docker exec ion-manager ion_ping_peers 1 2 3
   ```

Chapter 3

Product Support

There are two levels of support for the ANMS: troubleshooting by the administrator or user attempting to install or operate the ANMS, which is detailed in Section 3.1, and upstream support via the ANMS public GitHub project, accessible as described in Section 3.2. Attempts to troubleshoot should be made before submitting issue tickets to the upstream project.

3.1 Troubleshooting

3.1.1 Installation

This section covers issues that can occur during installation (see Section 2.2) of the ANMS.

3.1.1.1 SELinux Blocked Behavior

If there are errors related to the SELinux modules for the ANMS containers during installation of the ANMS on the local host, as discussed in Section 2.2, add the following line to the Puppet `common.yaml` file, typically found at `puppet/data/common.yaml`, and redeploy.

```
selinux::mode: permissive
```

This will result in the host being in permissive mode which allows activity not defined in SELinux modules but records those events to the system audit log. See Section 2.6.2 for details on observing the audit log events.

Caution
The SELinux permissive mode is for troubleshooting only and must not be present in a production installation.

3.1.2 Operations

This section covers issues that can occur after successful installation (see Section 2.2) and checkout (see Section 2.7) of the ANMS.

3.1.2.1 Grafana Containers

If the Grafana panels in the `Monitor` tab displays `Connection was reset` errors, the Grafana container may not have started successfully.

Restart the container with `docker-compose up grafana` (run from within the `anms/` folder).

If restarting the container does not resolve the problem, and the Grafana startup contains errors related to only having read-only access to the database, permissions on various files in the source code will need to be updated for Grafana to run.

For both the `docker_data/grafana_vol/` folder and the `docker_data/grafana_vol/grafana.db` file, change the group to `docker` and the permissions to `777`:

```
$ sudo chgrp docker docker_data/grafana__vol
$ sudo chgrp docker docker_data/grafana_vol/grafana.db
$ sudo chmod 777 docker_data/grafana_vol
$ sudo chmod 777 docker_data/grafana_vol/grafana.db
```

After changing these permissions, run `docker-compose up grafana` again, and the Grafana container should start sucessfully.

3.1.2.2 Agent Registration Issues on ANMS Startup

If an Agent is not present in the `Agents` tab on start up, it is likely due to an error in one of the ION containers and their connection to the underlying database.

To resolve the issue, restart the ION containers using `docker-compose restart n1 n2`.

3.1.2.3 New Agent Registration Issues

If registering a new Agent does not result in an update to the displayed Agents in the ANMS Agent tab, check that it has been registered to the Manager via the nm-manager CLI. The nm-manager CLI is accessible from a terminal, and this check can be done using a command such as:

```
docker exec -it ion-manager journalctl -f --unit ion-nm-mgr
```

If the results confirm that the Agent is registered but it still does not show on the Agents tab of the ANMS, there may be an issue with connection between the Manager and ANMS database.

This can be manually resolved by adding the Agent via the adminer DB tool that is deployed as part of the docker-compose tool at http://localhost/. The connection information is described in Section 3.1.2.4.

3.1.2.4 AMP Database Querying

To see what is present in the underlying AMP database, use the adminer access point.

With ANMS running, go to `localhost:8080` and log in to the database with: - System: `PostgreSQL` - Server: `postgres` - Username: `root` - Password: `root` - Database `amp_core`

3.1.2.5 ANMS-UI is not visible at hostname:9030

This error may indicate that the anms-ui docker is experiencing issues receiving HTTP requests. This is most likely related to the `host` or `bind address` specified in `anms-ui/server/shared/config.py`, or an environment variable that overrides this.

3.1.2.6 ANMS-UI is not visible at hostname

If http://hostname:9030 (replace hostname with the server's hostname) displays the ANMS UI, but http://hostname does not render the same page, this indicates an issue with NGinx.

Check the status of NGix in the docker-compose services list. It may be necessary to restart nginx via `docker-compose -f docker-compose.yml restart nginx`.

If this restart does not resolve the issue, check `nginx.conf` in the root of the ammos-anms project. Ensure that `anms-ui` or `localhost` are set to port `80` and the hostname is correct.

3.2 Contacting or Contributing

The ANMS is hosted on a GitHub repository [anms-source] with submodule references to several other repositories. There is a `CONTRIBUTING.md` document in the ANMS repository which describes detailed procedures for submitting tickets to identify defects and suggest enhancements.

Separate from the source for the ANMS proper, the ANMS Product Guide and User Guide are hosted on a GitHub repository [anms-docs], with its own `CONTRIBUTING.md` document for submitting tickets about either the Product Guide or User Guide.

While the GitHub repositories are the primary means by which users should submit detailed tickets, other inquiries can be made directly via email to the the support address dtnma-support@jhuapl.edu.

Index

A
AAA, 4
ADM, 6
Agent, 6

D
Data Warehouse, 6

F
FIPS-140, 6

P
Puppet, 3

R
RHEL, 3

S
SELinux, 6

U
UDPCL, 6