Watchmaker Documentation

Release 0.28.2

Plus3 IT Systems

Oct 31, 2023

CONTENTS

Applied Configuration Management

CHAPTER

ONE

OVERVIEW

Watchmaker is intended to help provision a system from its initial installation to its final configuration. It was inspired by a desire to eliminate static system images with embedded configuration settings (e.g. gold disks) and the pain associated with maintaining them.

Watchmaker works as a sort of task runner. It consists of "*managers*" and "*workers*". A *manager* implements common methods for multiple platforms (Linux, Windows, etc). A *worker* exposes functionality to a user that helps bootstrap and configure the system. *Managers* are primarily internal constructs; *workers* expose configuration artifacts to users. Watchmaker then uses a common *configuration file* to determine what *workers* to execute on each platform.

CHAPTER

TWO

CONTENTS

2.1 Installation

2.1.1 From Python Package Index

The preferred method to install `watchmaker` is from the Python Package Index (PyPi), using pip. Without any other options, this will always install the most recent stable release.

```
python3 -m pip install watchmaker
```

If you do not have Python or pip, this Python installation guide can guide you through the process.

Note: Versions 10 and later of `pip` do not support Python 2.6. On CentOS 6 and RHEL 6, Python 2.6 is the system version of Python. If you are using Python 2.6 with `watchmaker`, you will need to restrict the `pip` install such that a version earlier than 10 is installed. See the relevant question in the [FAQ](faq.html) for more details.

2.1.2 From source

Watchmaker can also be built and installed from source, using `git` and `pip`. The source for `watchmaker` are available from the GitHub repo.

1. First clone the public repository to pull the code to your local machine:

   ```
   git clone https://github.com/plus3it/watchmaker.git --recursive && cd watchmaker
   ```

   This project uses submodules, so it's easiest to use the `--recursive` flag, as above. If you don't, you will need to pull in the submodules as well:

   ```
   git submodule update --init --recursive
   ```

2. If you want to install a specific branch or tag, check it out before installing Watchmaker:

   ```
   git checkout <branch-tag-foo>
   ```

3. Then you can install Watchmaker:

```
python3 -m pip install .
```

2.1.3 From standalone executable

Watchmaker can also be downloaded and executed in an all-in-one package containing Watchmaker's dependencies, such as Python and necessary Python packages. Packages are available for Windows and Linux.

1. Retrieve the Watchmaker standalone package for your desired platform from GitHub Releases or the Cloudarmor repo.
   - GitHub Releases shows the available Watchmaker versions and includes links to the Windows and Linux packages, and their SHA256 hashes.
   - The latest release can also be directly accessed on GitHub:
     - `https://github.com/plus3it/watchmaker/releases/latest/`
   - The Cloudarmor repo also contains versioned Watchmaker packages and corresponding SHA256 hashes. You can browse the repo, or construct the URL to the files using these patterns:
     - `https://watchmaker.cloudarmor.io/releases/${VERSION}/watchmaker-${VERSION}-standalone-linux-x86_64`
     - `https://watchmaker.cloudarmor.io/releases/${VERSION}/watchmaker-${VERSION}-sha256-linux-x86_64.json`
     - `https://watchmaker.cloudarmor.io/releases/${VERSION}/watchmaker-${VERSION}-standalone-windows-amd64.exe`
     - `https://watchmaker.cloudarmor.io/releases/${VERSION}/watchmaker-${VERSION}-sha256-windows-amd64.json`
   - The latest release is always available on the Cloudarmor repo at these URLs:
     - `https://watchmaker.cloudarmor.io/releases/latest/watchmaker-latest-standalone-linux-x86_64`
     - `https://watchmaker.cloudarmor.io/releases/latest/watchmaker-latest-sha256-linux-x86_64.json`
     - `https://watchmaker.cloudarmor.io/releases/latest/watchmaker-latest-standalone-windows-amd64.exe`
     - `https://watchmaker.cloudarmor.io/releases/latest/watchmaker-latest-sha256-windows-amd64.json`
   - From PowerShell, the Windows package can be downloaded as follows:

```
PS C:\wam> $url = "https://watchmaker.cloudarmor.io/releases/latest/watchmaker-
↪latest-standalone-windows-amd64.exe"
PS C:\wam> (New-Object System.Net.WebClient).DownloadFile($url, "watchmaker.exe
↪")
```

   - From the command line, the Linux package can be downloaded as follows:

```
# curl -so watchmaker https://watchmaker.cloudarmor.io/releases/latest/
↪watchmaker-latest-standalone-linux-x86_64
```

   - For the latest package, the version of Watchmaker can be determined by viewing the contents of the SHA256 hash file or by executing the package with the `--version` flag.

2. Verify the integrity of the standalone package.

   Compare the SHA256 hash contained in the downloaded hash file to a hash you compute for the package.

   For Linux, execute this command to compute the SHA256 hash:

   ```
   # shasum -a 256 watchmaker-latest-standalone-linux-x86_64
   ```

   For Windows, execute this command to compute the SHA256 hash:

   ```
   PS C:\wam> Get-FileHash watchmaker-latest-standalone-windows-amd64.exe | Format-List
   ```

3. Set executable access permission.

   For Linux, you will need to set the access permissions to allow the standalone executable to run. Below is an example:

   ```
   # chmod +x watchmaker-latest-standalone-linux-x86_64
   ```

2.1.4 Prerequisites for features specific to AWS and Azure

Watchmaker has some features specific to AWS and Azure:

```
* AWS:
    * Download files in config references from Amazon S3
    * Tag Amazon EC2 instances with Watchmaker status
* Azure:
    * Tag Azure Virtual Machines with Watchmaker status
```

If you are using the source install from PyPi, and if your config uses any of those features, be sure to also install the SDKs those features are built on. If you are using the standalone package, these dependencies are part of the package and no further action or install is needed.

For AWS features, install the `boto3` library:

```
# python3 pip -m install boto3
```

For Azure features, install the `azure` libraries:

```
# python3 pip -m install azure-core azure-identity azure-mgmt-resource
```

2.2 Configuration

Watchmaker is configured using a YAML file. Watchmaker's default config.yaml file should work out-of-the-box for most systems and environments. You can also use it as an example to create your own configuration file. The default config file will install Salt and use the bundled Salt formulas to harden the system according to the DISA STIG.

The configuration is a dictionary. The parent nodes (keys) are: `all`, `linux`, or `windows`. The parent nodes contain a list of workers to execute, and each worker contains parameters specific to that worker. The `all` node is applied to every system, and `linux` and `windows` are applied only to their respective systems.

You can create a file using the above format with your own set of standard values and use that file for Watchmaker. Pass the CLI parameter `--config` to point to that file.

2.2.1 Configuration Precedence

In addition to passing values in the configuration file, watchmaker supports passing arguments on the *cli*. The order of precedence for arguments is, from least to most:

- configuration file
- cli argument

In other words, providing a value as a cli argument will override the same value provided in the configuration file.

2.2.2 config.yaml Parent Nodes

watchmaker_version

If used, this optional node constrains the version of Watchmaker that can be used with the configuration. The `watchmaker_version` node is recommended for all configurations used with versions of Watchmaker 0.17+.

This is an example of using the `watchmaker_version` node:

```
watchmaker_version: "== 0.17.0"
```

Any PEP440-compatible version specifier can be used in the `watchmaker_version` node. Each version clause should include a comparison operator, such as ~=, ==, !=, <=, >=, <, >, or ===. Multiple clauses can be included, separated by commas. Below are examples of version specifiers.

```
watchmaker_version: "~= 0.17.0"
watchmaker_version: "> 0.16.5"
watchmaker_version: ">= 0.17.0, <= 0.18.9, != 0.17.2"
```

Attempting to use a configuration with an incompatible version of Watchmaker will result in an error.

all

Section for Worker configurations that affect the deployment of all platforms. The `all` section will override parameters that are set in the OS-specific sections of config.yaml.

linux

Section for Worker configurations that should only be applied to Linux-based systems.

windows

Section for Worker configurations that should only be applied to Windows-based systems.

status

Watchmaker supports posting the watchmaker status to status providers. Watchmaker status values are one of: 'Running', 'Failed', or 'Completed'. Each status provider defines what it means to "post the status". Currently, the supported provider types include: 'aws' and 'azure'. These status providers both post the status as a tag to the instance/VM.

Providers have the ability to detect whether the system is compatible with the provider type. In order to post status, the system running watchmaker must be compatible with the status provider type. For example, the 'azure' provider will be skipped when watchmaker is running on an AWS EC2 instance, and vice versa.

See the *installation* page for prerequisites for using this feature.

- `IAM Role and Policy` An AWS Role and Policy that allows the instance to create tags must be attached to the instance. The minimal policy below has been tested in commercial and govcloud.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "arn:<PARTITION>:ec2:<REGION>:<ACCOUNT_ID>:instance/$
↪{ec2:InstanceID}",
            "Condition": {
                "StringLike": {
                    "ec2:SourceInstanceARN": "arn:<PARTITION>:ec2:<REGION>:<ACCOUNT_ID>
↪:instance/${ec2:InstanceID}"
                }
            }
        }
    ]
}
```

- `Policy` Policy that allows adding or replacing tag on resource see Microsoft Azure Tag Policy for more info.

Note: Note: Support for the 'azure' status provider is provisional. If you use it and encounter problems, please open an issue on the GitHub repository!

Parameters supported by status

- `providers` ((list of maps)): List of providers
  - `key` (*string*): Status key to use e.g. `WatchmakerStatus`
  - `required` (*boolean*): Required status, when `True` and provider_type is detected, watchmaker raises an error if unable to update status
  - `provider_type` (*string*): Environment provider type e.g. `aws` or `azure`

  Example:

```
status:
  providers:
    - key: 'WatchmakerStatus'
      required: False
      provider_type: 'aws'
    - key: 'WatchmakerStatus'
      required: False
      provider_type: 'azure'
```

2.2.3 Config.yaml Worker Nodes

Watchmaker includes the *workers* listed below. See the corresponding sections for details on their configuration parameters.

- *salt*
- *yum (linux-only)*

salt worker

Parameters supported by the Salt Worker:

- `admin_groups` (*list*): The group(s) that you would like the admin accounts placed within.
- `admin_users` (*list*): The user(s) that would be created as admins.
- `computer_name` (*string*): The computer or hostname that should be applied.
- `computer_name_pattern` (*string*): The pattern the `computer_name` value must match to be considered a valid name. Pattern is used to validate computer name and fail fast on invalid names. In addition, this value will be set as the salt grain `name-computer:pattern`, to be used by the name-computer formula.
- `environment` (*string*): Set for the environment in which the system is being built.
- `valid_environments` (*list*): The list of environments considered valid for the environment parameter.
- `ou_path` (*string*): Specifies the full DN of the OU where the computer account will be created when joining a domain.

```
ou_path: "OU=Super Cool App,DC=example,DC=com"
```

- `pip_install` (*list*): The Python package(s) that formulas require.

```
pip_install:
    - hvac
    - numpy
```

- `pip_args` (*list*): Options to pass to salt `pip.install` when installing python packages. See the salt docs for all options.

```
linux:
  - salt:
      pip_args:
        - pre_releases=True
```

- `pip_index` (*string*): Base URL of Python Package Index.
- `salt_states` (*string, comma-separated*): User-defined salt states to apply.

```
salt_states: highstate,foo,bar
```

- `exclude_states` (*string, comma-separated*): States to exclude from execution of salt states.
- `user_formulas` (*dict*): Map of formula names and URLs to zip archives of salt formulas. These formulas will be downloaded, extracted, and added to the salt file roots. The zip archive must contain a top-level directory that, itself, contains the actual salt formula. To "overwrite" bundled submodule formulas, make sure the formula name matches the submodule name.

```
user_formulas:
  foo-formula: https://path/to/foo.zip
```

- `salt_debug_log` (*string*): Path to the debug logfile that salt will write to.
- `salt_content` (*string*): URL to the Salt content file that contains further configuration specific to the salt install.
- `salt_content_path` (*string*): The path within the Salt content file specified using `salt_content` where salt files are located. Can be used to provide the path within the archive file where the Salt configuration files are located.
- `install_method` (*string*): (Linux-only) The method used to install Salt. Currently supports: `yum`, `git`
- `bootstrap_source` (*string*): (Linux-only) URL to the salt bootstrap script. This is required if `install_method` is set to `git`.
- `git_repo` (*string*): (Linux-only) URL to the salt git repo. This is required if `install_method` is set to `git`.
- `salt_version` (*string*): (Linux-only) A git reference present in `git_repo`, such as a commit or a tag. If not specifid, the HEAD of the default branch will be used.
- `installer_url` (*string*): (Windows-only) URL to the Salt Minion installer for Windows.

yum worker (linux-only)

Parameters supported by the Yum Worker:

- `repo_map` (list of maps): There be dragons here! Please be careful making changes to the default config. Thoroughly test your configuration. The default config specifies yum repos that contain the salt-minion. If the default repos are not included, and the salt-minion is not available, the Salt Worker will fail. You can add repos here that you would like enabled, but be wary of removing the default repos. Each map must contain the following keys:
  - `dist` (*list*): Distributions that would install this repo. Some repos are supported by multiple distros. (Currently supported distros are redhat, centos, and amazon.)
  - `el_version` (_string_): The Enterprise Linux version for this repo, as in el6 or el7. Expected values are `'6'` or `'7'`.

- `url` (*string*): URL location of the repo file to be added to the system. This file will be copied to `/etc/yum.repos.d/`

  Example:

```
repo_map:
  - dist:
      - redhat
      - centos
    el_version: 6
    url: http://someplace.com/my.repo
```

2.2.4 Downloading config files from Amazon S3

Watchmaker has support for downloading files from Amazon S3. This is useful for implementations where parts of the Watchmaker config are hosted privately. In order to use this feature, be sure the necessary prerequisites are installed (see the *installation page*).

This feature simply uses the "S3 URL" of the object in the S3 bucket. Such URLs take the form: `s3://<bucket>/<key>`. For example, if you wanted to host a custom config and custom salt content, you could include the salt-content S3 URL in your Watchmaker config:

```
all:
  - salt:
      salt_content: s3://path/to/salt-content.zip
```

And then call Watchmaker on the CLI with the `--config` argument:

```
watchmaker --config s3://path/to/config.yaml
```

2.2.5 Example config.yaml

This example can be used to construct your own `config.yaml` file. The Cloudarmor repo provides yum repo definitions and installers for a few salt versions.

```
watchmaker_version: ">= 0.24.0.dev"
all:
  - salt:
      admin_groups: null
      admin_users: null
      computer_name: null
      environment: null
      ou_path: null
      salt_content: null
      salt_states: Highstate
      user_formulas:
        # To add extra formulas, specify them as a map of
        #    <formula_name>: <archive_url>
        # The <formula_name> is the name of the directory in the salt file_root
        # where the formula will be placed. The <archive_url> must be a zip
        # file, and the zip must contain a top-level directory that, itself,
        # contains the actual salt formula. To "overwrite" submodule formulas,
```

(continues on next page)

(continued from previous page)

```
        # make sure <formula_name> matches submodule names. E.g.:
        #ash-linux-formula: https://s3.amazonaws.com/salt-formulas/ash-linux-formula-
↪master.zip
        #scap-formula: https://s3.amazonaws.com/salt-formulas/scap-formula-master.zip

linux:
  - yum:
      repo_map:
        #SaltEL6:
        - dist:
            - redhat
            - centos
          el_version: 6
          url: https://watchmaker.cloudarmor.io/yum.defs/saltstack/salt/2019.2.8/salt-
↪reposync-el6.repo
        - dist: amazon
          el_version: 6
          url: https://watchmaker.cloudarmor.io/yum.defs/saltstack/salt/2019.2.8/salt-
↪reposync-amzn.repo
        #SaltEL7:
        - dist:
            - redhat
            - centos
          el_version: 7
          url: https://watchmaker.cloudarmor.io/yum.defs/saltstack/salt/3004.2/salt-
↪reposync-el7-python3.repo
  - salt:
      salt_debug_log: null
      install_method: yum
      bootstrap_source: null
      git_repo: null
      salt_version: null

windows:
  - salt:
      salt_debug_log: null
      installer_url: https://watchmaker.cloudarmor.io/repo/saltstack/salt/windows/Salt-
↪Minion-3004.2-1-Py3-AMD64-Setup.exe

status:
  providers:
    - key: 'WatchmakerStatus'
      required: False
      provider_type: 'aws'
    - key: 'WatchmakerStatus'
      required: False
      provider_type: 'azure'
```

2.3 Usage

2.3.1 `watchmaker` from the CLI

Once Watchmaker is *installed* and a *configuration file* has been created (or you have decided to use the default configuration), using Watchmaker as a CLI utility is as simple as executing `watchmaker`. Below is the output of `watchmaker --help`, showing the CLI options.

In addition to the below options, any setting supported by the *configfiguration file* can be passed on the CLI. Some settings from the configuration file are not listed in the `--help` output, which displays only the most frequently used options. To pass any such "hidden" CLI argument, just precede it with `--` and convert an underscore to a dash. For example, to pass the `salt_content` argument on the CLI, use `watchmaker <other options> --salt-content <content-url>`. Arguments passed on the CLI always override the corresponding setting in the configuration file (see *configuration* for precedence).

```
# watchmaker --help
Usage: watchmaker [OPTIONS]

  Entry point for Watchmaker cli.

Options:
  --version                       Show the version and exit.
  -c, --config TEXT               Path or URL to the config.yaml file.
  -l, --log-level [info|debug|critical|warning|error]
                                  Set the log level. Case-insensitive.
  -d, --log-dir DIRECTORY         Path to the directory where Watchmaker log
                                  files will be saved.
  -n, --no-reboot                 If this flag is not passed, Watchmaker will
                                  reboot the system upon success. This flag
                                  suppresses that behavior. Watchmaker
                                  suppresses the reboot automatically if it
                                  encounters a failure.
  -s, --salt-states TEXT          Comma-separated string of salt states to
                                  apply. A value of 'None' will not apply any
                                  salt states. A value of 'Highstate' will
                                  apply the salt highstate.
  -A, --admin-groups TEXT         Set a salt grain that specifies the domain
                                  groups that should have root privileges on
                                  Linux or admin privileges on Windows. Value
                                  must be a colon-separated string. E.g.
                                  "group1:group2"
  -a, --admin-users TEXT          Set a salt grain that specifies the domain
                                  users that should have root privileges on
                                  Linux or admin privileges on Windows. Value
                                  must be a colon-separated string. E.g.
                                  "user1:user2"
  -t, --computer-name TEXT        Set a salt grain that specifies the
                                  computername to apply to the system.
  -e, --env TEXT                  Set a salt grain that specifies the
                                  environment in which the system is being
                                  built. E.g. dev, test, or prod
  -p, --ou-path TEXT              Set a salt grain that specifies the full DN
                                  of the OU where the computer account will be
```

(continues on next page)

(continued from previous page)

```
                                created when joining a domain. E.g.
                                "OU=SuperCoolApp,DC=example,DC=com"
  --help                        Show this message and exit.
```

Note: The `-c`/`--config` switch supports the use of `s3://` URLs. However, in order for such URLs to be treated as valid, it will be necessary to include the `boto3` Python module: if leveraging userData for either Windows or Linux (as below), include it on the same `pip install` line used to install watchmaker; if executing interactively (or by other non userData means), ensure that the relevant system-preparation processes performed to install `watchmaker` also include installation of the `boto3` module prior to invoking the `watchmaker` utility. Failure to ensure presence of the `boto3` Python module when referencing `s3://` URIs will result in logged-failures similar to:

```
2023-06-22 14:26:59,192 [backoff][INFO ][4908]: Backing off urlopen_retry(...) for 0.6s␣
↪(urllib.error.URLError: <urlopen error unknown url type: s3>)
2023-06-22 14:26:59,803 [backoff][ERROR][4908]: Giving up urlopen_retry(...) after 5␣
↪tries (urllib.error.URLError: <urlopen error unknown url type: s3>)
2023-06-22 14:26:59,803 [watchmaker.config][CRITICAL][4908]: Could not read config file␣
↪from the provided value "s3://<BUKKIT>/<PREFIX>/config.yaml"! Check that the config is␣
↪available.
```

2.3.2 `watchmaker` as a standalone package (Beta feature)

Standalone packages are a beta feature and may not function in all environments.

Once a Watchmaker standalone executable has been *downloaded* and a *configuration file* has been created (or you have decided to use the default configuration), use Watchmaker similarly to the CLI utility.

For example, on Linux, you can view the CLI options (shown above) using the same flag.

```
# ./watchmaker --help
```

From Windows, similarly, execute Watchmaker by running it from the command line:

```
PS C:\wam> watchmaker.exe --help
```

2.3.3 `watchmaker` in AWS

`watchmaker` as EC2 userdata

Calling Watchmaker via EC2 userdata is a variation on using it as a CLI utility. The main difference is that you must account for installing Watchmaker first, as part of the userdata. Since the userdata syntax and dependency installation differ a bit on Linux and Windows, we provide methods for each as examples.

Note: The `pip` commands in the examples are a bit more complex than necessarily needed, depending on your use case. In these examples, we are taking into account limitations in FIPS support in the default PyPi repo. This way the same `pip` command works for all platforms.

Linux

For **Linux**, you must ensure `pip` is installed, and then you can install `watchmaker` from PyPi. After that, run `watchmaker` using any option available on the *CLI*. Here is an example:

```
#!/bin/sh
PYPI_URL=https://pypi.org/simple

# Setup terminal support for UTF-8
export LC_ALL=en_US.UTF-8
export LANG=en_US.UTF-8

# Install pip
python3 -m ensurepip

# Install setup dependencies
python3 -m pip install --index-url="$PYPI_URL" --upgrade pip setuptools

# Install Watchmaker
python3 -m pip install --index-url="$PYPI_URL" --upgrade watchmaker

# Run Watchmaker
watchmaker --log-level debug --log-dir=/var/log/watchmaker
```

Alternatively, cloud-config directives can also be used on **Linux**:

```
#cloud-config

runcmd:
  - |
    PYPI_URL=https://pypi.org/simple

    # Setup terminal support for UTF-8
    export LC_ALL=en_US.UTF-8
    export LANG=en_US.UTF-8

    # Install pip
    python3 -m ensurepip

    # Install setup dependencies
    python3 -m pip install --index-url="$PYPI_URL" --upgrade pip setuptools

    # Install Watchmaker
    python3 -m pip install --index-url="$PYPI_URL" --upgrade watchmaker

    # Run Watchmaker
    watchmaker --log-level debug --log-dir=/var/log/watchmaker
```

Windows

For **Windows**, the first step is to install Python. `Watchmaker` provides a simple bootstrap script to do that for you. After installing Python, install `watchmaker` using `pip` and then run it.

```
<powershell>
$BootstrapUrl = "https://watchmaker.cloudarmor.io/releases/latest/watchmaker-bootstrap.
↪ps1"
$PythonUrl = "https://www.python.org/ftp/python/3.10.11/python-3.10.11-amd64.exe"
$PypiUrl = "https://pypi.org/simple"

# Use TLS 1.2+
[Net.ServicePointManager]::SecurityProtocol = "Tls12, Tls13"

# Download bootstrap file
$BootstrapFile = "${Env:Temp}\$(${BootstrapUrl}.split('/')[-1])"
(New-Object System.Net.WebClient).DownloadFile("$BootstrapUrl", "$BootstrapFile")

# Install python
& "$BootstrapFile" -PythonUrl "$PythonUrl" -Verbose -ErrorAction Stop

# Install Watchmaker
python -m pip install --index-url="$PypiUrl" --upgrade pip setuptools
python -m pip install --index-url="$PypiUrl" --upgrade watchmaker

# Run Watchmaker
watchmaker --log-level debug --log-dir=C:\Watchmaker\Logs
</powershell>
```

`watchmaker` as a CloudFormation template

Watchmaker can be integrated into a CloudFormation template as well. This project provides a handful of CloudFormation templates that launch instances or create autoscaling groups, and that install and execute Watchmaker during the launch. These templates are intended as examples for you to modify and extend as you need.

Sometimes it is helpful to define the parameters for a template in a file, and pass those to CloudFormation along with the template. We call those "parameter maps", and provide one for each of the CFN templates.

Cloudformation templates

- Linux Autoscale Group
- Linux Instance
- Windows Autoscale Group
- Windows Instance

`watchmaker` in a Terraform module

Watchmaker can also be used with Terraform by utilizing the Watchmaker AWS Terraform modules and passing the required parameters.

Terraform Modules

- Linux Autoscale Group
- Linux Instance
- Windows Autoscale Group
- Windows Instance

Note: Each corresponding Terraform module and CloudFormation template are grouped together in the same directory.

The CloudFormation templates are integrated within their respective Terraform module, so they become deployable and manageable from within the Terraform cli.

Variables can be input interactively via the Terraform console or directly to the Terraform module. An example Terraform file that calls the `lx-autoscale` module is shown below.

```
provider "aws" {}

module "test-lx-instance" {
  source = "git::https://github.com/plus3it/terraform-aws-watchmaker//modules/lx-
↪instance/"

  Name      = "tf-watchmaker-lx-autoscale"
  AmiId     = "__AMIID__"
  AmiDistro = "__AMIDISTRO__"
}
```

Additional Watchmaker Terraform examples

- Linux Autoscale Example
- Linux Instance Example
- Windows Autoscale Example
- Windows Instance Example

2.3.4 watchmaker in Azure

watchmaker as Custom Script Extension

Custom Script Extension downloads and executes scripts on Azure virtual machines. For Linux, you run the bash script shown in the section on *Linux*. You can store the bash script in Azure Storage or a publicly available url (such as with S3). Then you execute the stored script with a command. For example, a JSON string could contain

```
{
  "fileUris": ["https://path-to-bash-script/run_watchmaker.sh"],
  "commandToExecute": "./run_watchmaker.sh"
}
```

These parameters can be passed in via Azure CLI or within a Resource Management Template. For more in-depth information, see Microsoft's documentation on Linux.

For Windows, you would execute a PowerShell script in a similar manner as for *Windows* (but without the powershell tags). Then you would have the following parameters:

```
{
  "fileUris": ["https://path-to-bash-script/run_watchmaker.ps1"],
  "commandToExecute": "powershell -ExecutionPolicy Unrestricted -File run_watchmaker.ps1"
}
```

For more in-depth information on using Custom Script Extension for Windows, see Microsoft's documentation on Windows.

2.3.5 watchmaker as a library

Watchmaker can also be used as a library, as part of another python application.

```
import watchmaker

arguments = watchmaker.Arguments()
arguments.config_path = None
arguments.no_reboot = False
arguments.salt_states = None

client = watchhmaker.Client(arguments)
client.install()
```

Note: This demonstrates only a few of the arguments that are available for the `watchmaker.Arguments()` object. For details on all arguments, see the *API Reference*.

2.4 Execution Customization

This document is intended to help both the `watchmaker` user-community and `watchmaker` developers and contributors better understand how to customize the execution of the `watchmaker` configuration-utility. This will be covered in the documents linked-to from the (below) "**Common Scenarios**" section.

2.4.1 Background

By default, `watchmaker` executes a standard set of configuration-tasks. The `watchmaker` utility primarily leverages SaltStack for these configuration-tasks.

The configuration-tasks, themselves, are grouped into sets of related tasks. Related tasks can be things like:

- Performing OS-hardening (e.g., applying STIGs)
- Joining a Linux or Windows host to an Active Directory domain
- Installing/configuring enterprise-mandated software (e.g., anti-virus or other security-tooling)
- etc.

These task-sets are delivered in the form of formulas. From the vendor documentation on formulas:

> Formulas are pre-written Salt States. They are as open-ended as Salt States themselves and can be used for tasks such as installing a package, configuring, and starting a service, setting up users or permissions, and many other common tasks.
>
> All official Salt Formulas are found as separate Git repositories in the "saltstack-formulas" organization on GitHub

The `watchmaker` project follows a similar convention. Formulae specifically authored to work under `watchmaker` can be found by visiting Plus3 IT's GitHub and querying for the substring, "-formula".

2.4.2 Critical Files

Customization-activities will be governed by two, main files: the watchmaker configuration file (a.k.a.,`config.yaml`) and the Salt content archive (a.k.a., `salt-content.zip`). Discussion of the files' contents are as follows:

Dissecting The `config.yaml` File

The stock config.yaml file has five top-level directives or directive-maps:

- `watchmaker_version`
- `all`
- `linux`
- `windows`
- `status`

These directives or directive-dictionaries are used to govern the overall behavior of watchmaker's execution. See the default config.yaml file for generic layout and exemplar-content.

The `watchmaker_version` Directive

This directive applies a compatibility-filter to the watchmaker execution. If the installed version of watchmaker doesn't meet the version-critera set by this line, watchmaker won't work with this file's content. It's assumed that any watchmaker version that does not match the version-criteria will not (properly) support the configuration directives. Normally, the version is set as "greater than or equal to" string.

The `all` Map

This map is used to supply default values to both saltstack and to specific SaltStack formulae. The map-keys most likely to be of interest will be:

- `valid_environments`
- `salt_content`
- `salt_states`
- `salt_version`
- `user_formulas`

The `valid_environments` List-Parameter

This list provides a list of names of "environments" that saltstack's site-customization behavior has been configured to deliver. This will typically be used by environments that wish to customize deployments on an environment-by-environment basis (e.g. where an organization's development, testing/integration and production environments might have different endpoints to contact for things like configuring authentication) and wish to have a single `config.yaml` file to be used across all valid deployment-environments.

Typical values will be `null`,`dev`, `test` and/or `prod`.

- Usage of `null` indicates no differentiation between environments – that the generic configuration should be applied. This can mean that all the environments leverage the same service-integration information or that each deployment-environment will be governed by a different `config.yaml` file.
- The others are shorthand for "development", "testing & integration" and "production", respectively.

The word "typical" was previously emphasized because *any* value (other than `null`) is supported so long as there is a correspondingingly-named content-hierarchy in the site's custom Salt-content archive file (see the `salt-content` dictionary-key in the next section). This content-hierarchy needs to exist within the Salt-content archive file at `./pillar/<ENVIRONMENT_NAME>` (e.g., a `dev` environment would have a corresponding `./pillar/dev` directory in the Salt-content archive file).

Note: The default environment that watchmaker will apply is specified using the `environment` parameter. In the example, this is set to `null` – meaning that a generic configuration will be applied. This value is overridden by requesting a specific environment-configuration by using either `-e <ENVIRONMENT>` or `--env <ENVIRONMENT>` flag and argument when invoking watchmaker (per the *Usage Guide*).

The `salt_content` String-Parameter

This string defines where Watchmaker should attempt to download any site-customization content from. If this value is the literal `null`, watchmaker will not attempt to download any site-customization content. Otherwise, a valid URI pointing to a customized Salt-content archive should be used. This URI can point to a locally-staged file, an HTTP/HTTPS URL or an S3 URI.

If using an S3 URI, a couple of further requirements apply:

- When installing watchmaker, it will be necessary for the `boto3` Python library to be installed
- The to-be-configured system must have read access to the specified S3 URI

By default, `watchmaker` will extract this archive-file's contents at `/srv/watchmaker/salt` (Linux) or `C:\Watchmaker\Salt\srv` (Windows) the `./` referenced elsewhere in this document will be relative to that extraction-location.

Note: See the *Salt Contents Archive File* document for a discussion on the contents and layout of this file.

The `salt_states` String-Parameter

This parameter is by Watchmaker to invoke SaltStack with the desired states selected for execution. The typical value for this parameter is `Highstate`. The `Highstate` value tells watchmaker to invoke SaltStack with the `Highstate` invoker rather than iterated-states invoker. Invoking Saltstack with the `Highstate` invoker will cause all available and activated formulas to be selected for execution.

The `salt_version` String-Parameter

The value for this parameter instructs watchmaker which version of the Saltstack software it should download – or, if the correct version is already installed, skip re-downloading or re-installing. This will correspond to the value returned when `salt-call --version` is executed (after the `watchmaker` utility has downloaded and installed SaltStack). See the watchmaker changelog for guidance on latest supported version of Saltstack.

The `user_formulas` Dictionary-Parameter

This dictionary-parameter usually has no content. However, if one wishes to customize watchmaker's execution either by adding further formulae to install or to override installtion of default-formulae's contents with newer content (e.g., when testing updates to standard formulae), this dictionary should be populated. The expected value will take the form of:

```
    <FORMULA_NAME>: <DOWNLOAD_URL>
```

- <FORMULA_NAME> will be used as the installation-location for the formula-contents into the `/srv/watchmaker/salt/formulas` (Linux) or `C:\Watchmaker\Salt\srv\formulas` (Windows) directories.
- <DOWNLOAD_URL> will be used as the location from which to download an archive of target formula's content. This content should be in the form of a ZIP archive. Most frequently, this will be the public download-URL of a GitHub branch's (or commit-ID's) ZIP-archived, but any archive-URL that watchmaker has permission to download *should* work

For example, if one is working on updates to the `ash-linux-formula` and has made those changes in a GitHub project, one would specify a value of:

```
    ash-linux-formula: https://github.com/<USER_ID>/<PROJECT_NAME>/archive/refs/heads/
↪<BRANCH_NAME>.zip
```

The above will cause the content normally loaded at `.../formulas/ash-linux-formula` to be replaced with the content unarchved from the `https://github.com/<USER_ID>/<PROJECT_NAME>/archive/refs/heads/<BRANCH_NAME>.zip` archive-URI.

Similarly, if one is working on a *new* formula, specifying:

```
    <NEW_FORMULA_NAME>: https://github.com/<USER_ID>/<PROJECT_NAME>/archive/refs/heads/
↪<BRANCH_NAME>.zip
```

Will cause the content-archive hosted at the specified GitHub URL to be unarcheved at the platform-appropriate `.../formulas/<NEW_FORMULA_NAME>` directory-path. It *should* also cause an appropriate update to the SaltStack minion-configuration[1]. If this fails to happen, there is most likely a formatting issue with your `user_formulas` declaration. The following is (a snippet of) how the declaration *should* look:

```
all:
  salt:
    [...elided...]
    user_formulas:
      <FORMULA_NAME>: <SOURCE_URL>
```

Note that the `<FORMULA_NAME>` line is indented two spaces from the `user_formulas` token. If improper indentation is used - for example:

```
all:
  salt:
    [...elided...]
    user_formulas:
    <FORMULA_NAME>: <SOURCE_URL>
```

Neither the `.../formulas/<NEW_FORMULA_NAME>` directory-path will be created nor the `minion` file updated.

The `linux` Map

This map contains two top-level keys: `yum` and `salt`. Both are also maps.

The `yum` map instructs watchmaker about where to fetch `yum/dnf` repository-definition files from. Using the `yum:repo_map`, watchmaker will perform a lookup using the `dist:<distribution>` value and `el_version` value to identify the download `url` from which to pull the appropriate `yum/dnf` repository-definition files from

---

Note: Currently, mappings for `Red Hat 7`, `CentOS 7`, `Alma Linux 8`, `CentOS 8 Stream`, `Oracle Linux 8`, `Red Hat 8` and `Rocky Linux 8` are defined. Further Enterprise Linux distributions may be supported by appropriate extension of this map, along with further modifcations to a few Saltstack formulae.

The `salt` map is generally not modified for customization or other activities.

[1] The minion-configuration file is found at `/opt/watchmaker/salt/minion` on Linux hosts and `C:\Watchmaker\Salt\conf\minion` on Windows hosts.

The `windows` Map

Similar to the `linux` map, the `windows` map instructs watchmaker about where to fetch the Salt-minion setup-executable for Windows from.

As with the `linux` map, the `salt` dictionary is generally not modified for customization or other activities.

The `status` Map

This map defines the "status" content that watchmaker will attempt to write as a tag to the watchmaker-managed host. Currently, only Amazon EC2s and Azure VMs are supported. Currently, this map defaults to a value of:

```
status:
  providers:
    - key: WatchmakerStatus
      required: false
      provider_type: aws
    - key: WatchmakerStatus
      required: false
      provider_type: azure
```

The watchmaker finish/status routines interpret the above to mean, "apply the tag named `WatchmakerStatus` to the managed-host and set an appropriate completion-status value[2], but do not exit with an error if the tagging-operation fails".

Salt Contents Archive File

The SallStack contents archive contains three main file-hierarchies

- `pillar`
- `states`
- `winrepo`

The `pillar` Directory-Tree

This directry-hierarchy contains the pillar-data that is used to govern the behavior of executed SaltStack states. If modifying SaltStack states' behavior from their defaults (or supplying mandatory parameters), place the modifications under this directory-hierarchy.

Any automation-formulae that make reference to Pillar data – typically such formulae will include a `pillar.example` or `pillar.example.yml` in their content-root directory – may have their behavior modified through content under this directory-tree. Typically, this directory-tree will contain a `common` directory-tree and a directory-tree for each supported deployment environment. Thus, if one is creating behavioral-controls for `dev`, `test` and `prod` *environments*, the `pillar` directory tree will contain:

[2] The completion-status tagger will set a value of either `Completed` or `Error`.

- *common directory*
- dev directory
- prod directory
- test directory

Each formula that should be executed for a given site and environment should reference each formula's previously-mentioned pillar.example or pillar.example.yml files. Relevant content should be placed into either the common directory's init.sls files or in the environment-specific init.sls files. See below discussions for which directories should be used for a given method of setting parameter values.

In addition, the top-level directory will contain a top.sls and (optionally) a map.jinja file. These files help Watchmaker's SaltStack components know what further content to execute.

The top.sls file

The top.sls file-content will typically look like (taken from the watchmaker-salt-content project's pillar/top.sls file):

```
base:
  'G@os_family:RedHat':
    - common.ash-linux
    - common.scap.elx

  'G@os_family:Windows':
    - common.ash-windows
    - common.netbanner
    - common.scap.windows
    - common.winrepo
```

What this content does is selects a base set of Saltstack pillar-data to read in. This makes execution-customizations of selected saltstack formulae available for those formulae to consume. In the base-scenario (shown above), SaltStack will use the executing system's os_family to select which pillar-content to read in. Reading in only platform-relevant content helps reduce the amount of content that Saltstack has to read in.

Note: This value can be queried-for/verified on a host with watchmaker installed by executing:

```
salt-call -c /opt/watchmaker/salt/ --output text grains.get os_family
```

On an Enterprise Linux system, this will produce:

```
local:  RedHat
```

If further execution-customization content is desired to be made available to executing formulae, it may be set up here.

The map.jinja file

The presence/use of a map.jinja file is optional in the pillar root-directory. Any map.jinja file within the pillar *hierarchy* is designed to facilitate the loading of pillar data. The primary reason for having a map.jinja within the pillar root-directory is if there's common pillar data that needs to be shared across environments. If there's a need for such shared content, the pillar subdirectories would reference the shared file at /map.jinja (typically, if they reference a map.jinja file, at all, they will reference their "local" version by pointing at map.jinja).

The common Directory-Tree

As the name suggest, this directory-tree contains parameter-values that should be the same across all configured environments. This directory *typically* contains one or more subdirectories. Subdirectories will one-for-one match with the common.<NAME> strings found in the top-level pillar directory's top.sls file. Each sub-directory will contain one or more .sls files that will contain the relevant, common-across-all-environments pillar (per-formula parameter value) data. Thus, if one has a .../pillar/top.sls that looks like (as shown in the top.sls file subsection):

```
base:
  'G@os_family:RedHat':
    - common.ash-linux
    - common.scap.elx

  'G@os_family:Windows':
    - common.ash-windows
    - common.netbanner
    - common.scap.windows
    - common.winrepo
```

The .../pillar/common subdirectory will need to have the subdirectories:

- ash-linux
- ash-windows
- netbanner
- scap[1]
- winrepo

Each of the above-listed subdirectories – with the exception of the scap subdirectory will have an init.sls file[2]. Each of these files will contain parameter-dictionaries that align with invoked formulae's pillar.example or pillar.example.yml files.

To illustrate behavior tailoring, use the .../pillar/common/ash-linux/init.sls file with contents similar to:

```
{%- set os = salt.grains.filter_by({
    'AlmaLinux': 'centos',
    'CentOS': 'centos',
    'CentOS Stream': 'centos',
    'OEL': 'ol',
    'RedHat': 'rhel',
    'Rocky': 'centos',
```

(continues on next page)

[1] In the case of the scap directory (and content that functions similarly), instead of an init.sls file, it will instead have .sls files named to match what's in the top.sls file. In the illustrated case, that means a elx.sls and a windows.sls file.

[2] Formulae whose pillar-data is platform-separated such as the scap directory's typically will not need an init.sls file as the per-platform .sls files will directly-referenced and subsume the init.sls file's functionality.

(continued from previous page)

```
}, grain='os') %}

ash-linux:
  lookup:
    scap-profile: stig
    scap-cpe: /root/scap/content/openscap/ssg-rhel{{ grains['osmajorrelease'] }}-cpe-
↪dictionary.xml
    scap-xccdf: /root/scap/content/openscap/ssg-{{ os }}{{ grains['osmajorrelease'] }}-
↪xccdf.xml
    scap-ds: /root/scap/content/openscap/ssg-{{ os }}{{ grains['osmajorrelease'] }}-ds.
↪xml
```

Then compare it to the ash-linux project's pillar.example file. The project's example file attempts to show available parameters whose values can be set/overridden and how they fit into the parameter-map's structure. In the above, the `ash-linux:lookup:scap-profile` parameter's value is set to `stig`. However, if one consult's the formula-project's `pillar.example` file, it's found that any of the values `stig-rhel7-server-gui-upstream`, `standard`, `pci-dss`, `C2S` or `common` are valid[3].

As such, if one wanted to make the ash-linux-formula automation use a hardening-profile other than `stig`, one could specify any of the values found in that `pillar.example` file (e.g., change `stig` to `pci-dss` to use the `pci-dss` hardening-profile, instead).

Similarly, if one wanted to change where relevant SCAP-content should be loaded from the `scap-cpe`, `scap-xccdf` and/or `scap-ds` values could all be modified.

The <environment> Directory-Tree(s)

The <environment> directory-trees work similarly to the `common` directory tree. The primary difference is focus. Where the `common` directory-tree sets broad-scope behaviors via pillar-variables' parameter/values, the <environment> directory-trees set more-limited scopes' behaviors. These directory-trees are intended to align with an infrastructure-as-code environment where an organization has multiple, similar environments that each have specific needs (e.g., to point to per-environment CSP endpoints, security-services servers, etc., install different sets of software or apply different security-benchmarks).

The structure for the <environment> directory-trees is much simpler than that for the `common` directory tree. There are no subdirectories under each <environment> directory, just a single `init.sls` file. These typically take the form of:

```
{%- load_yaml as os_families %}
RedHat:
  <FORMULA_1_NAME>:
    lookup:
      <var1>: <VALUE>
      <var2>: <VALUE>
      ...
      <varN>: <VALUE>
  ...
  <FORMULA_N_NAME>:
    lookup:
      <var1>: <VALUE>
```

(continues on next page)

[3] While efforts are made to keep the examples up to date and correct, it's possible that "valid" parameter-values listed in a given watchmaker-formula project's `pillar.example` file will become invalid over time. If one encounters incorrect or missing exmaple parameter content in a given formula project's `pillar.example` file, please open an issue against that project.

(continued from previous page)

```
      <var2>: <VALUE>
      ...
      <varN>: <VALUE>
Windows:
  <FORMULA_1_NAME>:
    lookup:
      <var1>: <VALUE>
      <var2>: <VALUE>
      ...
      <varN>: <VALUE>
  ...
  <FORMULA_N_NAME>:
    lookup:
      <var1>: <VALUE>
      <var2>: <VALUE>
      ...
      <varN>: <VALUE>
```

Each of the <VALUE>s listed above may be string, list, dictionary or map data-types. The data-type will be dictated by the needs of the formula and illustrated in the relevant formulae's `pillar.example` or `pillar.example.yml` files. For example, to configure the mcafee-agent-formula to properly configure Trellix to run on Windows and Linux hosts in the `prod` environment, one would have a `.../pillar/prod/init.sls` file that looked something like:

```
{%- load_yaml as os_families %}
RedHat:
  <FORMULA_1_NAME>:
    lookup:
      <var1>: <VALUE>
      <var2>: <VALUE>
      ...
      <varN>: <VALUE>
  trellix-agent:
    lookup:
      source: s3://enterprise-software/mcafee/mcafee-agent/5.7.9/install.sh
      source_hash: s3://enterprise-software/mcafee/mcafee-agent/5.7.9/install.sh.SHA512
      client_in_ports:
        - 5575
      client_out_ports:
        - 80
        - 443
  ...
  <FORMULA_N_NAME>:
    lookup:
      <var1>: <VALUE>
      <var2>: <VALUE>
      ...
      <varN>: <VALUE>
Windows:
  <FORMULA_1_NAME>:
    lookup:
      <var1>: <VALUE>
      <var2>: <VALUE>
```

(continues on next page)

(continued from previous page)

```
    ...
    <varN>: <VALUE>
trellix-agent:
  lookup:
    version: '5.7.9.139'
  winrepo:
    versions:
      '5.7.9.139':
        installer: s3://enterprise-software/mcafee/mcafee-agent/5.7.9/FramePkg.exe
...
<FORMULA_N_NAME>:
  lookup:
    <var1>: <VALUE>
    <var2>: <VALUE>
    ...
    <varN>: <VALUE>
```

The above would instruct the mcafee-agent-formula (see the automation's pillar.example file for insight) automation to:

- Linux: Download and execute the the Trellix 5.7.9 installer from the `s3://enterprise-software/mcafee/mcafee-agent/5.7.9/` bucket-path – and validate the file's integrity using the checksum file `install.sh.SHA512` from the same S3 bucket-path – and set up firewalld inbound exceptions for port tcp/5575 and outbound exceptions for ports 80/tcp and 443/tcp.
- Windows: Download and execute the Trellix "Frame" 5.7.9.139 package's executable-installer from the `s3://enterprise-software/mcafee/mcafee-agent/5.7.9/` bucket-path.

Similar `init.sls` content would be needed for any other Watchmaker formula used to install, configure or manage the execution of software.

The `states` Directory-Tree

This directory-hierarchy governs *which* Saltstack states will be executed from the available SaltStack formulae. Typically, only modification to this directory's `top.sls` is needed:

- States that are not desired for execution can be commented out or wholly removed.
- States that require conditional-execution can be placed inside of appropriate (Jinja) condition-blocks
- States that are beyond what's defined in the default `top.sls` can be added here to ensure their execution during a full run of watchmaker
- If a change in execution-order is desired, alter the list-order: listed states are executed serially in first-to-last order

The `winrepo` Directory-Tree

This directory-hierarchy contains windows-specific automation-content. Unlike the `pillar` and `states` directory-trees, content in this directory-tree is not expected to be multi-platform.

2.4.3 Common Scenarios

The behavior of watchmaker can be easily customized towards several ends. The most-commonly encountered are:

Modifying Formulae Execution-Parameters

The `watchmaker` utility bundles several SaltStack formulae. These bundled-formulae's behaviors are, in turn, governed, by a set of Pillar-data that are also bundled with the `watchmaker`utility: pillar-data is how SaltStack states' behaviors may be modified. Sites that wish to either override the bundled-formulae's default behaviors or wish to run additional SaltStack formulae that are *not* in the default formula-bundle – and need to provide supporting behvior-tailoring – can do so by creating custom pillar-data.

Reference

When customizing Pillar content, it will also be necessary to use a site-specific, YAML-formatted configuration-file. Per the "*Usage*" document's "from the CLI" section, this file is *typically* named `config.yaml`. Any file name can be used so long as it matches what is passed via the `-c`/`--config` argument to the `watchmaker` utility. Further, this configuration-file may be specified as hosted on the local filesystem, any HTTP/HTTPS URL or an S3-hosted URI.

The `watchmaker` pillar-data is delivered by way of a ZIP-formatted content-archive. While this archive-file typically takes the name `salt-content.zip`, any filename may be used so long as it's properly referenced in the `watchmaker` configuration-file's `salt_content` directive (see the *`config.yaml` discussion* for a deeper dive into this file's contents, including a discussion of the `salt_content` parameter). The following exmaple configuration-file – with `salt_content` directive highlighted – is taken from watchmaker project:

```
watchmaker_version: ">= 0.27.2.dev"

all:
  - salt:
      admin_groups: null
      admin_users: null
      computer_name: null
      environment: null
      ou_path: null
      salt_content: null
      salt_states: Highstate
      salt_version: '3006.1'
      user_formulas:
        # To add extra formulas, specify them as a map of
        #    <formula_name>: <archive_url>
        # The <formula_name> is the name of the directory in the salt file_root
        # where the formula will be placed. The <archive_url> must be a zip
        # file, and the zip must contain a top-level directory that, itself,
        # contains the actual salt formula. To "overwrite" submodule formulas,
        # make sure <formula_name> matches submodule names. E.g.:
        #ash-linux-formula: https://s3.amazonaws.com/salt-formulas/ash-linux-formula-
↪master.zip
```

(continues on next page)

(continued from previous page)

```
        #scap-formula: https://s3.amazonaws.com/salt-formulas/scap-formula-master.zip

linux:
  - yum:
      repo_map:
        #SaltEL7:
        - dist:
            - redhat
            - centos
          el_version: 7
          url: https://watchmaker.cloudarmor.io/yum.defs/saltstack/salt/3006.1/salt-
↪reposync-el7-onedir.repo
        #SaltEL8:
        - dist:
            - almalinux
            - centos
            - oracle
            - redhat
            - rocky
          el_version: 8
          url: https://watchmaker.cloudarmor.io/yum.defs/saltstack/salt/3006.1/salt-
↪reposync-el8-onedir.repo
  - salt:
      pip_install:
        - dnspython
      salt_debug_log: null
      install_method: yum
      bootstrap_source: null
      git_repo: null
      salt_version: null

windows:
  - salt:
      salt_debug_log: null
      installer_url: https://watchmaker.cloudarmor.io/repo/saltstack/salt/windows/Salt-
↪Minion-3006.1-Py3-AMD64-Setup.exe

status:
  providers:
    - key: "WatchmakerStatus"
      required: False
      provider_type: "aws"
    - key: "WatchmakerStatus"
      required: False
      provider_type: "azure"
```

Note: If creating a new config-file for customizing your site's `watchmaker`-execution, it's recommended that config-file content *not* be copied from this document but from the `watchmaker` project, directly.

As with the configuration-file (passed via the `-c`/`--config` argument to the `watchmaker` utility), this file may be specified as hosted on the local filesystem, any HTTP/HTTPS URL or an S3-hosted URI.

Site-Specific Parameters

To implement localized-behavior with watchmaker, it will be necessary to change the `salt_content` paramter's value from `null` to the location of a SaltStack content-bundle. As mentioned previously, the content-bundle should be delivered in the form of a ZIP-formatted content-archive.

The the overall structure and format of the archive-bundle is discussed in the Salt Contents Archive File document. Site-specific *parameters* – and associated values – would be handled within the archive-bundle's *Pillar* data contents.

Bundle locations

Watchmaker currently supports pulling the Saltstack content-bundles from three types of locations: HTTP(S) server, S3 bucket or filesystem-path. The `salt_content` paramter's value is stated as a URI-notation. See the following subsections for guidance on location-specification.

It's worth noting that Watchmaker has not been tested to (directly) support accessing CIFS- or NFS-based network-shares. If it is desired to access a content-bundle from such a hosting-location, it is recommended to include share-mounting steps in any pre-Watchmaker execution-steps. Once the network-share is mounted, then watchmaker can access the content-bundle as though it was a locally-staged bundle (see below).

S3-Hosted Bundle

An S3-hosted bundle would be specified like:

```
s3://<BUCKET_NAME>/<FILE_PATH_PREFIX>/<ARCHIVE_FILE_NAME>
```

For example, "`s3://my-site-bukkit/watchmaker/salt-content.zip`"

Note: For S3-hosted URIs, it will be necessary to have ensured that the Python Boto3 modules have been installed prior to executing watchmaker

---

Webserver-Hosted Bundle

A bundle hosted on an HTTP server would be specified like:

```
https://<WEB_SERVER_FQDN>/<FILE_PATH_PREFIX>/<ARCHIVE_FILE_NAME>
```

For example, "`https://wamstuff.my-site.local/watchmaker/salt-content.zip`"

Note: Either HTTP or TLS-encrypted HTTP URIs are supported.

If potentially-sensitive data will be contained in the site-localization archive-file, it is recommended that access to this file be restricted. This can typically be done with authorized IP-blocks, API tokens or other "simple" authentication credentials. If this limitation comes in the form of an API token or a simple-auth credential, it will be necessary to specify the token or credentials as part of the HTTP URI.

Locally-staged Bundle

A locally-staged bundle (presumably downloaded and placed as part of a previously-executed launch-time automation-task) would be specified like:

```
file:///<FILESYSTEM_PATH_PREFIX>/<ARCHIVE_FILE_NAME>
```

For example, "`file:///var/tmp/watchmaker/salt-content.zip`"

Modifying List of Executed-Formulas to Meet Site-needs

The `watchmaker` utility bundles several SaltStack formulae. Which formulae are executed, in what order and under what conditions are governed by the `.../states/top.sls` file:

- On Linux systems, the value of `.../` will be `/srv/watchmaker/salt`
- On Windows systems, the value of `.../` will be `C:\Watchmaker\Salt\srv`

A typical `.../states/top.sls` will look something like:

```
{%- set environments = ['dev', 'test', 'prod', 'dx'] %}

base:
  'G@os_family:RedHat':
    - name-computer
    - scap.content
    - ash-linux.vendor
    - ash-linux.stig
    - ash-linux.iavm
    - scap.scan

  'G@os_family:Windows':
    - name-computer
    - pshelp
    - netbanner.custom
    - ash-windows.stig
    - ash-windows.iavm
    - ash-windows.delta
    - scap.scan
    - ash-windows.custom
```

Adding, removing or re-ordering entries in this list modifies which formulae watchmaker executes and in what order it executes them

Adding "Extra" Formulae to the Execution-List

In order to add a new formula to Wachmaker's execution-list, edit the `.../states/top.sls` file. For cross-platform formulae, ensure appropriate entries exist under both the `base:G@os_family:RedHat` and `base:G@os_family:Winodws` lists. To add a formula to the execution list, insert the formula-name into the list just as the already-configured formulae are. For example, to add the [cribl-agent-formula] to the RedHat execution, modify the above RedHat stanza to look like:

```
  'G@os_family:RedHat':
    - name-computer
    - scap.content
    - ash-linux.vendor
    - ash-linux.stig
    - ash-linux.iavm
    - cribl-agent
    - scap.scan
```

If there are any futher conditionals that should be placed on the formula being added, surround the target-formula's list entry with suitable, Jinja-based conditional-operators. For example, if you want to ensure that the `cribl-agent` is executed when a suitable environment-value is specified, update the preceeding example to look like:

```
  'G@os_family:RedHat':
    - name-computer
    - scap.content
    - ash-linux.vendor
    - ash-linux.stig
    - ash-linux.iavm
{%- if salt.grains.get('watchmaker:enterprise_environment') | lower in environments %}
    - cribl-agent
{%- endif %}
    - scap.scan
```

Removing Formulae the Execution-List

In order to prevent a formula from being automatically run by Watchmaker, edit the `.../states/top.sls` file and either wholly remove the referenced-formula from the list or comment it out. To make the `scap-formula`'s `scan` state not run[1], modify the example `.../states/top.sls` file to look like:

```
base:
  'G@os_family:RedHat':
    - name-computer
    - scap.content
    - ash-linux.vendor
    - ash-linux.stig
    - ash-linux.iavm
```

or:

```
  'G@os_family:RedHat':
    - name-computer
```

(continues on next page)

[1] This is often done by system-owners that value launch-time provisioning-automation speed over the presence of an intial hardening-scan report on the launched-systems hard drive.

(continued from previous page)

```
    - scap.content
    - ash-linux.vendor
    - ash-linux.stig
    - ash-linux.iavm
##    - scap.scan
```

Changing Formulae the Execution-Order

There may be times where the system-owner will want Watchmaker to run formulae in a different order than previously-configured. The `.../states/top.sls` specifies formulae and states' execution-order serially. The order is top-to-bottom (with items closer to the top of the list executed earlier and those closer to the bottom of the lis executed later). To change the order that formulae are executed, change the order of the execution-list.

```
{%- set environments = ['dev', 'test', 'prod', 'dx'] %}

base:
  'G@os_family:RedHat':
    - name-computer
    - ash-linux.vendor
    - ash-linux.stig
    - ash-linux.iavm
    - scap.content
    - scap.scan

  'G@os_family:Windows':
    - name-computer
    - pshelp
    - netbanner.custom
    - ash-windows.stig
    - ash-windows.iavm
    - ash-windows.delta
    - ash-windows.custom
    - scap.scan
```

In the above (when compared to the `.../states/top.sls` file near the top of this document), the Linux `scap.content` formula-state and the Windows `scap.scan` formula-state have been moved to a later executions. This is an atypical change, but is provided for completeness' sake.

Note: Some times, particularly when creating new states, it is dicsovered that some SaltStack formulae or states are not as idempotent as they were intended to be. Re-ordering the executions may work around issues caused by an insufficient degree of idempotency in one or more formulae.

It is generally recommended, if idempotency-issues require execution-orders be modified, that the insufficiently-idempotent SaltStack formulae or states be refactored to improve their idempotency.

Testing Updates to Existing Formulas

The formulae-contents that are installed and configured for use by Watchmaker can be modified through a custom `config.yaml` file. This is done through the `config.yaml` file's `user_formula` dictionary-parameter (see: the *discussion* of the `config.yaml` file's `user_formulas` parameter and *take special note of guidance around file-formatting and indenting*). This parameter may be used to enable the setup of "in progress" updates to existing formulae. This is done by specifying dictionary-values for `user_formulas` in a custom `config.yaml` file:

```
all:
  salt:
    [...elided...]
    user_formulas:
      <TEST_FORMULA_1>: <TEST_FORMULA_ARCHIVE_1_URI>
      <TEST_FORMULA_2>: <TEST_FORMULA_ARCHIVE_2_URI>
      [...elided...]
      <TEST_FORMULA_N>: <TEST_FORMULA_ARCHIVE_N_URI>
```

Note: While multiple formulae are shown in the above snippet, it's not generally recommended to use this method for more than one, formula at a time. The above is primarily to illustrate that the `user_formulas` parameter is a dictionary.

Once the custom `config.yaml` file is in the desired state, it can be uploaded to an S3-based testing-bucket, web server[1] or even staged locally within the testing-system.

About Testing Updates

To the greatest extent possible, formulae should be *portable*. It is recommended that when testing updates, the developer:

- Tests with a bog-standard configuration (the custom `config.yaml` file's `salt_content` parameter's value is set to `null`)
- Tests with the target-environment's or target-environments' custom `salt-content.zip` file(s)
- Tests with a customized version of the bog-standard `salt-config.zip` file if to-be-tested formula's config-inputs have been changed
- Tests with a `salt-config.zip` file cloned from the target-environment's or target-environments' custom `salt-config.zip` file if to-be-tested formula's config-inputs have been changed

Exercising across environments, in this way, will better assure that proposed updates do not break an existing formula's portability.

[1] If hosting on a web server and configuration content may be deemed sensitive, apply suitable access controls to the file and specify the fetch-URL with the appropriate authentication-elements.

About Hosting of Modified Formula-Content

The modified formulae's contents *can* be installed from any Watchmaker-supported source-type – S3-hosted, web server hosted or local files. However, it is expected that most personnel attempting to test modifications to existing formulae will want to load that modified content *directly* from their development content-management system (CMS). To have watchmaker load content *directly* from the source CMS:

1. Visit the CMS (GitHub.Com when devloping for the main Watchmaker project)
2. Navigate to the content-developer's source fork/branch
3. Find the `https://` URL of fork/branch's ZIP-archive of the code to be tested.
4. Use the value from the prior step as the value for `<TEST_FORMULA_ARCHIVE_URI>`

Modification of existing/already-integrated formulae's content typically takes place on GitHub.Com. As of this document's authoring date, the above process looks like:

1. Browse to `https://www.github.com/plus3it/<FORMULA_NAME>`
2. Click on the down-arrow on the `Fork` button to bring up the `Existing Forks` list
3. Click on the developer's fork
4. On the landing-page for the developer's fork, click on the branch button's down-arrow[2]. This opens the `Switch branches/tags` dropdown
5. Select the desired branch from the `Switch branches/tags` dropdown
6. Click on the `Code` button's down-arrow. Right-click on the `Download ZIP` text in the drop-down that the down-arrow opens. This opens a context-menu of link-actions.
7. Click on the `Copy Link Address` menu-option (if using a browser other than Chrome, the specific menu-option may be different, but the equivalent action should be obvious) to copy the branch-archive's URL into your system's copy-buffer.
8. Paste the branch-archive's URL (from your system's copy-buffer) as the `<TEST_FORMULA_ARCHIVE_URI>` value in your custom `config.yaml` file

> **Warning:** If you created your modification-branch in a private fork, it will be necessary to create an API-token that grants your test-host the ability to access the archive-URL. Creating such tokens is outside the scope of this document.

Execution With Standard Configuration-Options

Assuming that the executing system has access to the specified URI(s), watchmaker will:

1. Download the requested formula ZIP-archive(s)
2. Unarchive them to the `.../formulas` directory – replacing the standard contents with the testing-contents

As an already-integrated formula, it should already be executed when a full/generic Watchmaker-run is requested.

While the modified formula should execute in place of the already-integrated formula contents as part of a full/generic Watchmaker-run, it will save testing-time to execute *only* the modified formula. This can be done by explicitly selecting *only* the modified-formula for execution using a method similar to:

- Linux invocation:

[2] This button will typically start out labeled either `master` or `main` (depending how old the formula's project is)

```
watchmaker \
  -c s3://<TESTING_BUCKET>/config.yaml \
  -s <FORMULA_NAME> \
  --log-level debug --log-dir=/var/log/watchmaker
```

- Windows invocation:

```
watchmaker --log-level debug --log-dir=C:\Watchmaker\Logs -c s3://<TESTING_BUCKET>/
↪config.yaml -s <FORMULA_NAME>
```

The modified formula's execution will be logged into the directory requested via the manual invocation.

Execution With Modified/New Configuration-Options

Because Watchmaker will overwrite existing formula-content with the referenced formula-content, it should only be necessary to execute the updated formula with a custom `salt-content.zip` if:

- One wishes to test with specific testing-values for existing formula-parameters
- The formula-updates add new parameters
- The formula-updates rename existing parameters
- The formula-updates change existing parameters' data-types.

If executing to cover one of the above scenarios, it will be necessary to either manually update the `.../pillar` directory's contents with the appropiate data (see: The `pillar` Directory-Tree) or create a custom `salt-config.zip` file and reference it from the custom `config.yaml` file.

Final Notes

If modification of an existing formula adds or removes parameters, renames existing parameters or changes existing parameters' data-types, it is *critical* that the formula-project's `pillar.example` or `pillar.example.yaml` file be updated to reflect these changes.

Testing New Formulas

The formulae-contents that are installed and configured for use by Watchmaker can be modified through a custom `config.yaml` file. This is done through the `config.yaml` file's `user_formula` dictionary-parameter (see: the *discussion* of the `config.yaml` file's `user_formulas` parameter and *take special note of guidance around file-formatting and indenting*). This parameter may be used to enable the setup of new, yet-to-be integrated formulae[1]. This is done by specifying dictionary-values for `user_formulas` in a custom `config.yaml` file:

[1] "Yet-to-be-integrated" formulae are any formulas that have not yet been set up for automated execution as part of a *full* Watchmaker run. See the Modifying Formulae Execution-Parameters document for tips.

```
all:
  salt:
    [...elided...]
    user_formulas:
      <NEW_FORMULA_1>: <NEW_FORMULA_ARCHIVE_1_URI>
      <NEW_FORMULA_2>: <NEW_FORMULA_ARCHIVE_2_URI>
      [...elided...]
      <NEW_FORMULA_N>: <NEW_FORMULA_ARCHIVE_N_URI>
```

Note: While multiple new formulae are shown in the above snippet, it's not generally recommended to use this method for more than one, new formula at a time. The above is primarily to illustrate that the *user_formulas* parameter is a dictionary

Once the custom `config.yaml` file is in the desired state, it can be uploaded to an S3-based testing-bucket, web server[2] or even staged locally within the testing-system.

About Testing New Formulae

To the greatest extent possible, formulae should be portable. It is recommended that when testing updates, the developer:

- Tests without use of a custom `salt-content.zip`
- Tests using custom Pillar-data – either by hand-modifying Pillar content or using a modified `salt-content.zip` cloned from the target deployment-environments' `salt-content.zip` – for one or more targeted deployment-environments

Exercising across environments, in this way, will better assure that newly-created formulae operate as portably as expected prior to the newly-created formulae's integration into standard or site-specific Watchmaker executions.

Execution - Generic/Defaults

Assuming that the executing system has access to the specified URIs, watchmaker will:

1. Download the requested formula ZIP-archive(s)
2. Unarchive them to the `.../formulas` directory
3. Update the `.../minion` file's `file_roots:base` list

If the site's `salt-content.zip` has not been modified to cause execution, the new formula can be explicitly executed using a method similar to:

- Linux invocation:

```
watchmaker \
  -c s3://<TESTING_BUCKET>/config.yaml \
  -s <FORMULA_NAME> \
  --log-level debug --log-dir=/var/log/watchmaker
```

- Windows invocation:

[2] If hosting on a web server and configuration content may be deemed sensitive, apply suitable access controls to the file and specify the fetch-URL with the appropriate authentication-elements.

```
watchmaker --log-level debug --log-dir=C:\Watchmaker\Logs -c s3://<TESTING_BUCKET>/
↪config.yaml -s <FORMULA_NAME>
```

The new formula's execution will be logged into the directory requested via the manual invocation.

Execution - Tailored

If the new formula has variable configuration-data that needs to come from pillar, it will be necessary to either manually update the `.../pillar` directory's contents with the appropiate data (see: The `pillar` Directory-Tree) or create a custom `salt-config.zip` file and reference it from the custom `config.yaml` file.

Final Notes

All formulae that have Pillar-settable or Pillar-overridable parameters should include a `pillar.example` or `pillar.example.yaml` file with the project's content. This file should be placed in the project's root-directory. The file should be valid YAML with explanatory comments for each configuration item. If a new formula's execution-customizability is more complex than is easily accommodated by comment-entries in the example Pillar YAML file, add a `README_PillarContents.md` file to the project. This file should contain sufficiently-expository content to allow new users of the formula to fully understand how to tailor the formulae's execution to their site's needs.

If there are other customization-scenarios that should be included in this document-set, please see the *contribution guidance*. The contribution document covers how to submit requests for documentation-improvements as well as guidance on how to contribute changes (like further customization documentation).

2.5 Troubleshooting Guidance

Troubleshooting Watchmaker activities can be done by checking various system logs. Logfile locations vary by OS and *may* vary by OS-version and cloud-provider. The per-OS, logfile discussions assume that you have executed Watchmaker per the relevant OSes' direct-usage guidance:

2.5.1 Linux Log-Files

The logfiles to pay most attention to when running Watchmaker on Enterprise Linux distros (Red Hat, CentOS, Oracle Enterprise, etc.) are as follows:

The `/var/log/watchmaker/watchmaker.log` Log-File

This file tracks the top-level execution of the `watchmaker` configuration-utility. This file should always exist, unless:

- The provisioning-administrator has checked for the log before the utility has been downloaded and an execution-attempted. This typically happens if a `watchmaker`-execution is attempted late in a complex provisioning-process
- An execution-attempt wholly failed. In this case, check the logs for the `watchmaker`-calling service or process (e.g. *`cloud-init`*)
- The provisioning-administrator has not invoked `watchmaker` in accordance with the `watchmaker` project's usage-guidance: if a different logging-location was specified (e.g., by adding a flag/argument like `--log-dir=/tmp/watchmaker`), the provisioning-administrator would need to check the alternately-specified logging-location.
- The provisioning-administrator invoked the `watchmaker`-managed *content* directly (e.g., using `salt-call -c /srv/watchmaker/salt`). In this scenario, only the content-execution may have been logged (whether logging was captured and where would depend on how the direct-execution was requested).

Typical Errors

- Bad specification of remotely-hosted configuration file. This will typically come with an HTTP 404 error similar to:

```
botocore.exceptions.ClientError: An error occurred (404) when calling the␣
↪HeadObject operation: Not Found
```

  Ensure that the requested URI for the remotely-hosted configuration file is valid.

- Attempt to use a protected, remotely-hosted configuration-file. This will typically come win an HTTP 403 error. Most typically, this happens when the requested configuration-file exists on a protected network share and the requesting-process doesn't have permission to access it.

```
botocore.exceptions.ClientError: An error occurred (403) when calling the␣
↪HeadObject operation: Forbidden
```

  Ensure that `watchmaker` has adequate permissions to access the requested, remotely-hosted configuration file.

- Remotely-hosted configuration file is specified as an `s3://` URI without installation of `boto3` Python module. This will typically come with an error similar to:

```
2023-06-22 14:26:59,192 [backoff][INFO ][4908]: Backing off urlopen_retry(...) for␣
↪0.6s (urllib.error.URLError: <urlopen error unknown url type: s3>)
2023-06-22 14:26:59,803 [backoff][ERROR][4908]: Giving up urlopen_retry(...) after␣
```

(continues on next page)

(continued from previous page)

```
↪5 tries (urllib.error.URLError: <urlopen error unknown url type: s3>)
2023-06-22 14:26:59,803 [watchmaker.config][CRITICAL][4908]: Could not read config␣
↪file from the provided value "s3://<BUKKIT>/<PREFIX>/config.yaml"! Check that the␣
↪config is available.
```

Ensure that the `boto3` Python module has been installed *prior to* attempting to execute `watchmaker`

The `/var/log/watchmaker/salt_call.debug.log` Log-File

This is the log-file that captures the bulk of the SaltStack-related state-output. This file gets created when `watchmaker` has been able to successfully download all of its execution information. This file gets created shortly after this line appears in the `/var/log/watchmaker/watchmaker.log` file:

```
2023-06-15 11:13:27,378 [watchmaker.workers.base.SaltLinux][DEBUG][6407]: Command: /usr/
↪bin/salt-call --local --retcode-passthrough --no-color --config-dir /opt/watchmaker/
↪salt --log-file /var/log/watchmaker/salt_call.debug.log --log-file-level debug --log-
↪level error --out quiet --return local state.highstate
```

Typically, the only errors that will appear here are the results of errors in the SaltStack formulae for the standard integrations. To see which modules *may* get logged into this file, look at the contents of the `/srv/watchmaker/salt/formulas/` directory and then cross-reference those directories against the contents of the `/srv/watchmaker/salt/states/top.sls` file. To help interpret, a typical `top.sls` file's contents is offered:

```
{%- set environments = ['dev', 'test', 'prod', 'dx'] %}

base:
  'G@os_family:RedHat':
    - name-computer
    - scap.content
    - ash-linux.vendor
    - ash-linux.stig
    - ash-linux.iavm
{%- if salt.grains.get('watchmaker:enterprise_environment') | lower in environments %}
    - join-domain
    - mcafee-agent
    - splunkforwarder
    - nessus-agent.elx.install
    # Recommend other custom states be inserted here
{%- endif %}
    - scap.scan

  'G@os_family:Windows':
    [...elided...]
```

In the above, these salt formulas will be executed unconditionally on RedHat-derivative systems:

- `/srv/watchmaker/salt/formulas/name-computer-formula`

- /srv/watchmaker/salt/formulas/ash-linux-formula[1]

Similarly, the contents of the following directories will be executed by watchmaker only if the environment specified in the watchmaker-invocation (the string-value after the -e flag) matches one of the elements in the environments list.

- /srv/watchmaker/salt/formulas/join-domain-formula
- /srv/watchmaker/salt/formulas/mcafee-agent-formula
- /srv/watchmaker/salt/formulas/nessus-agent-formula
- /srv/watchmaker/salt/formulas/splunkforwarder-formula

Similarly, the behavior of each of the above states' executions will be governed by content specified under the /srv/watchmaker/salt/pillar directory hierarchy. This content is used to feed values into the parameter-driven SaltStack states enumerated in the .../formulas directories.

Typical Error Causes

The most frequent causes of errors, once watchmaker has caused Saltstack states to begin their execution, are errors encountered while running the individual enterprise-integration states. Typically, these errors are around stale configuration data (expired domain-join credentials for directory-integration or stale host/IP/port information for other services) or communication-issues between the OS that watchmaker is configuring and the service watchmaker is attempting to configure the instance to integrate: DNS resolution, host or network-level firewall rules, other transit-issues.

The next most frequent errors are already-existing configuration problems in the OS that watchmaker is configuring. These include things like:

- Failures accessing RPM repositories (especially problematic with repositories that require client-cert authentication where there are certificate-expiration problems between the RPM client and repository server)
- Too little storage in critical partitions
- The watchmaker activities running after something else has changed a resource-configuration that watchmaker expects to manage but finds the resource in an unanticipated state

The least frequent cause of errors is related to the SaltStack code itself. Usually, this is caught in pre-release testing, but "bugs happen". While states are typically coded to try to gracefully handle errors encountered – they'll typically still fail, but at least try to provide meaningful error-output. Usually, the "bugs happen" errors are resultant of environment-to-environment deltas that were not adequately specified to the code-maintainers or the requisite logic-branching was not able to be adequately exercised across the various environments.

For errors in enterprise-integration content, efforts have been undertaken to try to ensure those errors are adequately represented in this log-file. However, the application-specific logs (the ones *for* the integrated-application) will still remain the authoritative source for troubleshooting exercises.

[1] Due to the ash-linux.vendor, ash-linux.stig and ash-linux.iavm specification, only the ash-linux-formula's vendor, stig and iavm states' executions will be attempted.

The /var/log/messages Log-File

This is Red Hat Enterprise Linux's default/primary logging location for miscellaneous system activities. Any init- or systemd-launched service that emits output to STDERR or STDOUT will typically (also) log to this file.[1]

Typically, the provisioning-administrator will wish to review this file to trace where failures in the invocation of watchmaker have failed or where errors in an instance's/VM's userData payload has encountered errors.

- Search, case-insensitively, for the string "`watchmaker`" to find logged-content explicit to the execution of watchmaker. Depending how far watchmaker got before failing, there can be a significant amount of output to pore through (recommend piping to a pagination-tool such as `less`)
- Search for the string "`\ cloud-init:\ `" to find logged-content related to the `cloud-init` service. This search-string will reveal execution-output made to STDOUT and STDERR by any processes initiated by the `cloud-init` service. This will *typically* include watchmaker and any logging-enabled userData script-output. Search output will tend to be even more-significant than looking just for `watchmaker` (therefore, also recommend piping to a pagination-tool such as `less`)

The use of the qualifier, "typically", in the prior bullet is required to account for different methods for invoking `watchmaker`. Some `watchmaker`-users leverage methods such as CloudFormation and other templating-engines, Ansible and other externalized provisioning-services, etc. to launch the `watchmaker` process. Those methods are outside the scope of this document. The relevant logging should be known to the user of these alternate execution-frameworks.

The /var/log/cloud-init.log Log-File

This is the default location where the Red Hat packaged version of the `cloud-init` service for Enterprise Linux 6 and 7 writes *all* of its log-output to – on RHEL 8+, logging data is split-out across this file and the `/var/log/cloud-init-output.log` file. All automation directly-initiated through `cloud-init` and that emits STDOUT and/or STDERR messages will be duplicated here.

Primary diagnostic-use with respect to execution of `watchmaker` will be in tracking errors emitted during *preparation to execute* `watchmaker`. If the `watchmaker` process fails to start (meaning that `/var/log/watchmaker/watchmaker.log` is never created), this would be a good location to find *why* `watchmaker` failed to start.

Useful string-searches for locating executional points-of-interest ("landmarks") will be (ordered most- to least-useful):

- `:  FAIL:`
- `/var/lib/cloud/instance/script`
- `/var/lib/cloud/instance`
- `:  SUCCESS:`

By far, the search for `:  FAIL:` will be the most important in uncovering errors. The other searches will mostly be of use in progress-tracking and verifying expected event-sequencing[1].

[1] Some sites will explicitly disable local logging to this file. If this has been done, data that normally shows up in `/var/log/messages` *may*, instead, be found in the systemd output logs. See the Using journald document for a fuller detailing of using `journald` logging.

[1] Event-sequencing issues most-frequently happen when a userData payload delivers two or more scripts. When multiple scripts are specified in a userData payload, they are not necessarily executed in the same order they're specified in the userData text-stream. Instead, `cloud-init` executes scripts placed into the `/var/lib/cloud/instance/scripts/` directory in alphabetical order. Thus, if one *needs* the scripts to execute in a specific order, it is important to carefully name them such that that happens (e.g., `00_script` and `01_script` would result in the `00_`-prefixed script executing prior the `01_`-prefixed script)

Example Failure

Typically, searching for ": FAIL: will bring the file-cursor to a logged-block similar to:

```
2023-06-21 11:12:36,078 - subp.py[DEBUG]: Unexpected error while running command.
Command: ['/var/lib/cloud/instance/scripts/00_script.sh']
Exit code: 1
Reason: -
Stdout: -
Stderr: -
2023-06-21 11:12:36,078 - cc_scripts_user.py[WARNING]: Failed to run module scripts-user␣
↪(scripts in /var/lib/cloud/instance/scripts)
2023-06-21 11:12:36,078 - handlers.py[DEBUG]: finish: modules-final/config-scripts-user:␣
↪FAIL: running config-scripts-user with frequency once-per-instance
2023-06-21 11:12:36,078 - util.py[WARNING]: Running module scripts-user (<module
↪'cloudinit.config.cc_scripts_user' from '/usr/lib/python3.6/site-packages/cloudinit/
↪config/cc_scripts_user.py'>) failed
2023-06-21 11:12:36,079 - util.py[DEBUG]: Running module scripts-user (<module
↪'cloudinit.config.cc_scripts_user' from '/usr/lib/python3.6/site-packages/cloudinit/
↪config/cc_scripts_user.py'>) failed
Traceback (most recent call last):
  File "/usr/lib/python3.6/site-packages/cloudinit/stages.py", line 1090, in _run_modules
    run_name, mod.handle, func_args, freq=freq
  File "/usr/lib/python3.6/site-packages/cloudinit/cloud.py", line 55, in run
    return self._runners.run(name, functor, args, freq, clear_on_fail)
  File "/usr/lib/python3.6/site-packages/cloudinit/helpers.py", line 185, in run
    results = functor(*args)
  File "/usr/lib/python3.6/site-packages/cloudinit/config/cc_scripts_user.py", line 44,␣
↪in handle
    subp.runparts(runparts_path)
  File "/usr/lib/python3.6/site-packages/cloudinit/subp.py", line 426, in runparts
    % (len(failed), ",".join(failed), len(attempted))
RuntimeError: Runparts: 1 failures (00_script.sh) in 1 attempted commands
```

In this case, the failure happened during the execution of the userdata-script, `/var/lib/cloud/instance/scripts/00_script.sh`. Even if the script hasn't logged anything directly useful in this log file or hasn't even been configured to log its own activities any where, knowing that it was during the execution of this file is useful.

1. The provisioning-administrator knows where in the `cloud-init` automation-sequence things failed
2. One can look in other logs for actionable diagnostic-information
3. If there's no such information in other log files, one can hand-execute the failing script to see if the error can be reproduced (and in a way that assists the provisioning-administator with isolating the source of the failure)

For the third point, if the failure is in a BASH script, executing the script with the diagnostic flag set (e.g., `bash -x /var/lib/cloud/instance/scripts/00_script.sh`) one may be able to see where the script fails.

Similarly, if hand-execution of the script *succeeds* it can point to the script making incorrect assumptions about the `cloud-init` managed execution environment. This can include things like:

- Lack of necessary environment variables
- Improperly defined environment-variables
- Attempts to execute commands that require a controlling-TTY (i.e., an interactive-login shell)

- Attempting to do something that the instance's security posture blocks[2].

Note that comparing execution via `cloud-init` versus execution from an interactive-shell works whether the script is written in BASH or some other interpreted language.

The `/var/log/cloud-init-output.log` Log-File

This is the default location where the Red Hat packaged version of the `cloud-init` service for Enterprise Linux 8 and higher writes its *summary* log-output to. Primary content of potential troubleshooting-interest that can get logged here is the output from userData scripts.

The above are specifed in the order most-frequently used to determine execution issues.

Note that the troubleshooting discussions assume that `watchmaker` execution has been effected directly through the `cloud-init` service. If `watchmaker` is being executed by other means, the above files may have no relevance to issues encountered running `watchmaker` (the `cloud-init.log` and `cloud-init-output.log`), may not exist in the documented-locations (`salt_call.debug.log` and `watchmaker.log`) and may not even exist at all (`watchmaker.log`).

2.5.2 Windows Log-Files

When using `watchmaker` on Windows servers, the primary log-files of interest are:

The `c:\watchmaker\logs\watchmaker.log` Log-File

This file tracks the top-level execution of the watchmaker configuration-utility. This file should *always* exist. The primary reasons that it may not exist are:

- The provisioning-administrator has checked for the log before the `watchmaker`-utility has been downloaded and an execution-attempted. This typically happens if a `watchmaker`-execution is attempted late in a complex provisioning-process
- An execution-attempt wholly failed. In this case, check the logs for the watchmaker-calling service or process.

[2] SELinux can be especially problematic for processes started by `cloud-init`. For example, the `firewall-cmd` utility is not directly usable. `cloud-init` scripts would need to either issue a `setenforce 0` before invoking the command or use the alternate `firewall-offline-command`

- The provisioning-administrator has not invoked `watchmaker` in accordance with the `watchmaker` project's usage-guidance: if a different logging-location was specified (e.g., by adding a flag/argument like `--log-dir=C:\TEMP\watchmaker`), the provisioning-administrator would need to check the alternately-specified logging-location.
- The provisioning-administrator invoked the `watchmaker`-managed content directly (e.g., using `salt-call -c c:\watchmaker\salt\conf state.highstate`). In this scenario, only the content-execution may have been logged (whether logging was captured and where would depend on how the direct-execution was requested).

Location Note

The cited-location of the main `watchmaker`-execution's log-file is predicated on the assumption that `watchmaker` has been executed per the Usage-guidance for *Windows*:

```
<powershell>
$BootstrapUrl = "https://watchmaker.cloudarmor.io/releases/latest/watchmaker-bootstrap.
↪ps1"
$PythonUrl = "https://www.python.org/ftp/python/3.10.11/python-3.10.11-amd64.exe"
$PypiUrl = "https://pypi.org/simple"

# Use TLS 1.2+
[Net.ServicePointManager]::SecurityProtocol = "Tls12, Tls13"

# Download bootstrap file
$BootstrapFile = "${Env:Temp}\$(${BootstrapUrl}.split('/')[-1])"
(New-Object System.Net.WebClient).DownloadFile("$BootstrapUrl", "$BootstrapFile")

# Install python
& "$BootstrapFile" -PythonUrl "$PythonUrl" -Verbose -ErrorAction Stop

# Install Watchmaker
python -m pip install --index-url="$PypiUrl" --upgrade pip setuptools
python -m pip install --index-url="$PypiUrl" --upgrade watchmaker

# Run Watchmaker
watchmaker --log-level debug --log-dir=C:\Watchmaker\Logs
</powershell>
```

The value of the `--log-dir` parameter sets the directory-location where `watchmaker` will create its log-files, including the `watchmaker.log` file. If a different value is set for the `--log-dir` parameter, the log-file will be created in *that* directory-location, instead.

Typical Errors

- Bad specification of remotely-hosted configuration file. This will typically come with an HTTP 404 error similar to:

  ```
  botocore.exceptions.ClientError: An error occurred (404) when calling the␣
  ↪HeadObject operation: Not Found
  ```

  Ensure that the requested URI for the remotely-hosted configuration file is valid.

- Attempt to use a protected, remotely-hosted configuration-file. This will typically come win an HTTP 403 error. Most typically, this happens when the requested configuration-file exists on a protected network share and the requesting-process doesn't have permission to access it.

```
botocore.exceptions.ClientError: An error occurred (403) when calling the␣
↪HeadObject operation: Forbidden
```

  Ensure that `watchmaker` has adequate permissions to access the requested, remotely-hosted configuration file.

- Remotely-hosted configuration file is specified as an `s3://` URI without installation of `boto3` Python module. This will typically come with an error similar to:

```
2023-06-22 14:26:59,192 [backoff][INFO ][4908]: Backing off urlopen_retry(...) for␣
↪0.6s (urllib.error.URLError: <urlopen error unknown url type: s3>)
2023-06-22 14:26:59,803 [backoff][ERROR][4908]: Giving up urlopen_retry(...) after␣
↪5 tries (urllib.error.URLError: <urlopen error unknown url type: s3>)
2023-06-22 14:26:59,803 [watchmaker.config][CRITICAL][4908]: Could not read config␣
↪file from the provided value "s3://<BUKKIT>/<PREFIX>/config.yaml"! Check that the␣
↪config is available.
```

  Ensure that the `boto3` Python module has been installed *prior to* attempting to execute `watchmaker`

Alternate Logs

As noted above, this logfile may not exist if execution of watchmaker has wholly failed. If the execution was attempted via automated-startup methods but there is no watchmaker logfile, it will be necessary to check the CSP provider-logs. On AWS, the logs to check (per the vendor documentation) will be:

- If using (legacy) EC2Launch, the log-file to search will be *C:\ProgramData\Amazon\EC2-Windows\Launch\Log\UserdataExecution.log*
- If using EC2Launch v2, the log-file to search will be *C:\ProgramData\Amazon\EC2Launch\log\agent.log*

The `c:\watchmaker\logs\salt_call.debug` Log-File

This file captures the execution-activities of SaltStack formulae. This file will exist if `watchmaker` has be able to successfully download and install its (SaltStack-based) configuration-content.

The primary diagnostic interest in this file is if there is an execution-failure within a managed-content module. By default, `watchmaker` will reboot a system after a successful run[1]. If the expected reboot occurs, this file likely will not be of interest. If the reboot fails to occur and the `watchmaker` log indicates that it was able to start the SaltStack-based operations, then consult this file to identify what failed and (possibly) why.

[1] This behavior may be overridden by having invoked `watchmaker` with the `-n` flag

Typical Errors

Any errors encountered by SaltStack will typically have a corresponding log-section that starts with a string like:

```
2023-06-27 12:57:39,841 [salt.state       :325 ][ERROR   ][5656] { ... }
```

Errors from the failing SaltStack action will typically include an embedded JSON-stream. The above snippet's `{ ... }` stands in for an embedded JSON-stream (for brevity's sake). Depending how long the embedded JSON-stream is, it will probably make things easier for the provisioning-user to convert that stream to a more human-readable JSON document-block.

The most commonly-reported issues are around:

- Domain Join Errors

Domain Join Error

Errors in joining the host to active directory can have several causes. The three most typical are:

- Bad join-user credentials (or locked-out account)
- Inability to find domain controllers
- Inability to communicate with found domain controllers.

The following is version of the salt_call.debug log file with a join-domain failure. The version shown has the JSON-stream expanded into a (more-readable) JSON-document. The original content can be viewed to illustrate *why* expanding the JSON-stream makes the provisioning-administrator's life easier.

```
[...elided...]
2023-06-27 12:57:39,841 [salt.state       :325 ][ERROR   ][5656]
{
  'pid': 5420,
  'retcode': 1,
  'stdout': '
    VERBOSE: Performing the operation "Join in domain \'plus3it.lab\\aws-c2c4418931.
↪plus3it.lab\'" on target
    "ip-0A005598".\nWARNING: Command [xAdd-Computer] failed. Retrying in 10 second(s).\
↪nVERBOSE: Performing the operation "Join in domain \'plus3it.lab\\aws-c2c4418931.
↪plus3it.lab\'" on target
    "ip-0A005598".\nWARNING: Command [xAdd-Computer] failed. Retrying in 20 second(s).\
↪nVERBOSE: Performing the operation "Join in domain \'plus3it.lab\\aws-c2c4418931.
↪plus3it.lab\'" on target
    "ip-0A005598".\nWARNING:

    PSMessageDetails      :
    Exception             : System.Management.Automation.RuntimeException: Command [xAdd-
↪Computer] failed
    TargetObject          : Command [xAdd-Computer] failed
    CategoryInfo          : OperationStopped: (Command [xAdd-Computer] failed:String) [],
↪ RuntimeException
    FullyQualifiedErrorId : Command [xAdd-Computer] failed
    ErrorDetails          :
    InvocationInfo        : System.Management.Automation.InvocationInfo
    ScriptStackTrace      : at Retry-TestCommand, C:\\ProgramData\\Salt
```

(continues on next page)

(continued from previous page)

```
                            Project\\Salt\\var\\cache\\salt\\minion\\extfiles\\join-
↪domain\\JoinDomain.ps1: line 519
                            at <ScriptBlock>, C:\\ProgramData\\Salt
                            Project\\Salt\\var\\cache\\salt\\minion\\extfiles\\join-
↪domain\\JoinDomain.ps1: line 678
                            at <ScriptBlock>, <No file>: line 1
    PipelineIterationInfo : {}



    WARNING: Command [xAdd-Computer] failed the maximum number of 3 time(s).\n11/14/2018␣
↪16:05:56:270 ----------------------------------------------------------------------
    11/14/2018 16:05:56:270 NetpDoDomainJoin
    11/14/2018 16:05:56:270 NetpDoDomainJoin: using new computer names
    11/14/2018 16:05:56:270 NetpDoDomainJoin: NetpGetNewMachineName returned 0x0
    11/14/2018 16:05:56:270 NetpMachineValidToJoin: \'WIN-VDQFC8O4JEM\'
    11/14/2018 16:05:56:270 NetpMachineValidToJoin: status: 0x0
    11/14/2018 16:05:56:270 NetpJoinWorkgroup: joining computer \'WIN-VDQFC8O4JEM\' to␣
↪workgroup \'WORKGROUP\'
    11/14/2018 16:05:56:270 NetpValidateName: checking to see if \'WORKGROUP\' is valid␣
↪as type 2 name
    11/14/2018 16:05:56:286 NetpCheckNetBiosNameNotInUse for \'WORKGROUP\' [ Workgroup␣
↪as MACHINE]  returned 0x0
    11/14/2018 16:05:56:286 NetpValidateName: name \'WORKGROUP\' is valid for type 2
    11/14/2018 16:05:56:614 NetpJoinWorkgroup: status:  0x0
    11/14/2018 16:05:56:614 NetpDoDomainJoin: status: 0x0
    06/23/2023 13:59:59:151 ----------------------------------------------------------------
↪----
    06/23/2023 13:59:59:151 NetpDoDomainJoin
    06/23/2023 13:59:59:151 NetpDoDomainJoin: using new computer names
    06/23/2023 13:59:59:151 NetpDoDomainJoin: NetpGetNewMachineName returned 0x0
    06/23/2023 13:59:59:151 NetpDoDomainJoin: NetpGetNewHostName returned 0x0
    06/23/2023 13:59:59:151 NetpMachineValidToJoin: \'IP-0A005598\'
    06/23/2023 13:59:59:151         OS Version: 10.0
    06/23/2023 13:59:59:151         Build number: 17763 (17763.rs5_release.180914-1434)
    06/23/2023 13:59:59:151         SKU: Windows Server 2019 Datacenter
    06/23/2023 13:59:59:151         Architecture: 64-bit (AMD64)
    06/23/2023 13:59:59:245 NetpMachineValidToJoin: status: 0x0
    06/23/2023 13:59:59:245 NetpJoinDomain
    06/23/2023 13:59:59:245         HostName: ip-0A005598
    06/23/2023 13:59:59:245         NetbiosName: IP-0A005598
    06/23/2023 13:59:59:245         Domain: plus3it.lab\\aws-101f37cc48.plus3it.lab
    06/23/2023 13:59:59:245         MachineAccountOU: (NULL)
    06/23/2023 13:59:59:245         Account: dom-joiner
    06/23/2023 13:59:59:245         Options: 0x403
    06/23/2023 13:59:59:245 NetpDisableIDNEncoding: no domain dns available - IDN␣
↪encoding will NOT be disabled
    06/23/2023 13:59:59:245 NetpJoinDomainOnDs: NetpDisableIDNEncoding returned: 0x0
    06/23/2023 13:59:59:292 NetpJoinDomainOnDs: status of connecting to dc \'\\\\aws-
↪101f37cc48.plus3it.lab\': 0x0
    06/23/2023 13:59:59:307 NetpJoinDomainOnDs: Passed DC \'aws-101f37cc48.plus3it.lab\'␣
↪verified as DNS name \'\\\\aws-101f37cc48.plus3it.lab\'
```

(continues on next page)

(continued from previous page)

```
    06/23/2023 13:59:59:307 NetpDsGetDcName: status of verifying DNS A record name␣
↪resolution for \'aws-101f37cc48.plus3it.lab\': 0x0
    06/23/2023 13:59:59:307 NetpGetDnsHostName: PrimaryDnsSuffix defaulted to DNS domain␣
↪name: plus3it.lab
    06/23/2023 13:59:59:323 NetpProvisionComputerAccount:
    06/23/2023 13:59:59:323         lpDomain: plus3it.lab
    06/23/2023 13:59:59:323         lpHostName: ip-0A005598
    06/23/2023 13:59:59:323         lpMachineAccountOU: (NULL)
    06/23/2023 13:59:59:323         lpDcName: aws-101f37cc48.plus3it.lab
    06/23/2023 13:59:59:323         lpMachinePassword: (null)
    06/23/2023 13:59:59:323         lpAccount: dom-joiner
    06/23/2023 13:59:59:323         lpPassword: (non-null)
    06/23/2023 13:59:59:323         dwJoinOptions: 0x403
    06/23/2023 13:59:59:323         dwOptions: 0x40000003
    06/23/2023 13:59:59:370 NetpLdapBind: Verified minimum encryption strength on aws-
↪101f37cc48.plus3it.lab: 0x0
    06/23/2023 13:59:59:370 NetpLdapGetLsaPrimaryDomain: reading domain data
    06/23/2023 13:59:59:370 NetpGetNCData: Reading NC data
    06/23/2023 13:59:59:370 NetpGetDomainData: Lookup domain data for: DC=plus3it,DC=lab
    06/23/2023 13:59:59:370 NetpGetDomainData: Lookup crossref data for: CN=Partitions,
↪CN=Configuration,DC=plus3it,DC=lab
    06/23/2023 13:59:59:370 NetpLdapGetLsaPrimaryDomain: result of retrieving domain␣
↪data: 0x0
    06/23/2023 13:59:59:370 NetpGetLocalDACDisabled: returning 0x0, *pfDACDisabled=TRUE
    06/23/2023 13:59:59:370 NetpCheckForDomainSIDCollision: returning 0x0(0).
    06/23/2023 13:59:59:385 NetpGetComputerObjectDn: Cracking DNS domain name plus3it.
↪lab/ into Netbios on \\\\aws-101f37cc48.plus3it.lab
    06/23/2023 13:59:59:385 NetpGetComputerObjectDn: Crack results:         name =␣
↪PLUS3IT\\
    06/23/2023 13:59:59:385 NetpGetComputerObjectDn: Cracking account name PLUS3IT\\IP-
↪0A005598$ on \\\\aws-101f37cc48.plus3it.lab
    06/23/2023 13:59:59:385 NetpGetComputerObjectDn: Crack results:         Account does␣
↪not exist
    06/23/2023 13:59:59:385 NetpGetComputerObjectDn: Cracking Netbios domain name␣
↪PLUS3IT\\ into root DN on \\\\aws-101f37cc48.plus3it.lab
    06/23/2023 13:59:59:385 NetpGetComputerObjectDn: Crack results:         name =␣
↪DC=plus3it,DC=lab
    06/23/2023 13:59:59:385 NetpGetComputerObjectDn: Got DN CN=IP-0A005598,CN=Computers,
↪DC=plus3it,DC=lab from the default computer container
    06/23/2023 13:59:59:385 NetpGetADObjectOwnerAttributes: Looking up attributes for␣
↪machine account: CN=IP-0A005598,CN=Computers,DC=plus3it,DC=lab
    06/23/2023 13:59:59:385 NetpGetADObjectOwnerAttributes: Ldap Search failed: 8240
    06/23/2023 13:59:59:385 NetpCheckIfAccountShouldBeReused: Computer Object does not␣
↪exist in OU.
    06/23/2023 13:59:59:385 NetpCheckIfAccountShouldBeReused:fReuseAllowed: TRUE,␣
↪NetStatus:0x2030
    06/23/2023 13:59:59:385 NetpModifyComputerObjectInDs: Initial attribute values:
    06/23/2023 13:59:59:385                 objectClass  =  Computer
    06/23/2023 13:59:59:385                 SamAccountName  =  IP-0A005598$
    06/23/2023 13:59:59:385                 userAccountControl  =  0x1000
    06/23/2023 13:59:59:385                 DnsHostName  =  ip-0A005598.plus3it.lab
    06/23/2023 13:59:59:385                 ServicePrincipalName  =  HOST/ip-0A005598.
```

(continues on next page)

(continued from previous page)

```
↪plus3it.lab  RestrictedKrbHost/ip-0A005598.plus3it.lab  HOST/IP-0A005598 ␣
↪RestrictedKrbHost/IP-0A005598
    06/23/2023 13:59:59:385                 unicodePwd  =  <SomePassword>
    06/23/2023 13:59:59:385 NetpModifyComputerObjectInDs: Computer Object does not exist␣
↪in OU
    06/23/2023 13:59:59:385 NetpModifyComputerObjectInDs: Attribute values to set:
    06/23/2023 13:59:59:385                 objectClass  =  Computer
    06/23/2023 13:59:59:385                 SamAccountName  =  IP-0A005598$
    06/23/2023 13:59:59:385                 userAccountControl  =  0x1000
    06/23/2023 13:59:59:385                 DnsHostName  =  ip-0A005598.plus3it.lab
    06/23/2023 13:59:59:385                 ServicePrincipalName  =  HOST/ip-0A005598.
↪plus3it.lab  RestrictedKrbHost/ip-0A005598.plus3it.lab  HOST/IP-0A005598 ␣
↪RestrictedKrbHost/IP-0A005598
    06/23/2023 13:59:59:385                 unicodePwd  =  <SomePassword>
    06/23/2023 13:59:59:448 Querying "CN=IP-0A005598,CN=Computers,DC=plus3it,DC=lab" for␣
↪objectSid attribute
    06/23/2023 13:59:59:448 NetpQueryObjectSidAttribute succeeded: got RID=0x16c30␣
↪objectSid=S-1-5-21-3217479199-34324276-1494086650-93232
    06/23/2023 13:59:59:448 NetpDeleteMachineAccountKey: called for computer \'IP-
↪0A005598\'
    06/23/2023 13:59:59:464 NetpGetComputerObjectDn: Cracking DNS domain name plus3it.
↪lab/ into Netbios on \\\\aws-101f37cc48.plus3it.lab
    06/23/2023 13:59:59:464 NetpGetComputerObjectDn: Crack results:         name =␣
↪PLUS3IT\\
    06/23/2023 13:59:59:464 NetpGetComputerObjectDn: Cracking account name PLUS3IT\\IP-
↪0A005598$ on \\\\aws-101f37cc48.plus3it.lab
    06/23/2023 13:59:59:464 NetpGetComputerObjectDn: Crack results:         (Account␣
↪already exists) DN = CN=IP-0A005598,CN=Computers,DC=plus3it,DC=lab
    06/23/2023 13:59:59:464 NetpDeleteMachineAccountKey: msDS-KeyCredentialLink attr was␣
↪not found on computer \'IP-0A005598\' - no action required.
    06/23/2023 13:59:59:464 NetpDeleteMachineAccountKey: returning Status: 0
    06/23/2023 13:59:59:464 ldap_unbind status: 0x0
    06/23/2023 13:59:59:464 NetpJoinCreatePackagePart: status:0x0.
    06/23/2023 13:59:59:495 NetpJoinDomainOnDs: Setting netlogon cache.
    06/23/2023 13:59:59:495 NetpJoinDomainOnDs: status of setting netlogon cache: 0x0
    06/23/2023 13:59:59:495 NetpJoinDomainOnDs: Function exits with status of: 0x0
    06/23/2023 13:59:59:495 NetpJoinDomainOnDs: status of disconnecting from \'\\\\aws-
↪101f37cc48.plus3it.lab\': 0x0
    06/23/2023 13:59:59:495 NetpJoinDomain: DsrIsDeviceJoined returned false
    06/23/2023 13:59:59:620 NetpJoinDomain: NetpCompleteOfflineDomainJoin SUCCESS:␣
↪Requested a reboot :0x0
    06/23/2023 13:59:59:620 NetpDoDomainJoin: status: 0x0
     Setting backup/restore privileges.
    06/23/2023 13:59:59:464 NetpProvGetWindowsImageState: IMAGE_STATE_COMPLETE.
    06/23/2023 13:59:59:464 NetpAddPartCollectionToRegistry.
    06/23/2023 13:59:59:464 NetpProvGetTargetProductVersion: Target product version: 10.
↪0.17763.4252
    06/23/2023 13:59:59:479 NetpAddPartCollectionToRegistry: delete OP state key status:␣
↪0x2.
    06/23/2023 13:59:59:479 NetpConvertBlobToJoinState: Translating provisioning data to␣
↪internal format
    06/23/2023 13:59:59:479 NetpConvertBlobToJoinState: Selecting version 1
```

(continues on next page)

(continued from previous page)

```
    06/23/2023 13:59:59:479 NetpConvertBlobToJoinState: exiting: 0x0
    06/23/2023 13:59:59:495 NetpJoin2RequestPackagePartInstall: Successfully persisted␣
↪all fields
    06/23/2023 13:59:59:495 NetpJoin3RequestPackagePartInstall: Successfully persisted␣
↪all fields
    06/23/2023 13:59:59:495 NetpAddPartCollectionToRegistry: Successfully initiated␣
↪provisioning package installation: 3/3 part(s) installed.
    06/23/2023 13:59:59:495 NetpAddPartCollectionToRegistry: status: 0x0.
    06/23/2023 13:59:59:495 NetpOpenRegistry: status: 0x0.
    06/23/2023 13:59:59:495 NetpSetPrivileges: status: 0x0.
    06/23/2023 13:59:59:495 NetpRequestProvisioningPackageInstall: status: 0x0.
    06/23/2023 13:59:59:495 ----------------------------------------------------------------
↪----
    06/23/2023 13:59:59:495 NetpProvContinueProvisioningPackageInstall:
    06/23/2023 13:59:59:495         Context: 0
    06/23/2023 13:59:59:495 NetpProvGetWindowsImageState: IMAGE_STATE_COMPLETE.
    06/23/2023 13:59:59:510 NetpCreatePartListFromRegistry: status: 0x0.
    06/23/2023 13:59:59:510 NetpCompleteOfflineDomainJoin
    06/23/2023 13:59:59:510         fBootTimeCaller: FALSE
    06/23/2023 13:59:59:510         fSetLocalGroups: TRUE
    06/23/2023 13:59:59:510 NetpJoinDomainLocal: NetpHandleJoinedStateInfo returned: 0x0
    06/23/2023 13:59:59:588 NetpJoinDomainLocal: NetpManageMachineSecret returned: 0x0.
    06/23/2023 13:59:59:588 Calling NetpQueryService to get Netlogon service state.
    06/23/2023 13:59:59:588 NetpJoinDomainLocal: NetpQueryService returned: 0x0.
    06/23/2023 13:59:59:588 NetpJoinDomainLocal: status of setting LSA pri. domain: 0x0
    06/23/2023 13:59:59:588 NetpManageLocalGroupsForJoin: Adding groups for new domain,␣
↪removing groups from old domain, if any.
    06/23/2023 13:59:59:604 NetpManageLocalGroupsForJoin: status of modifying groups␣
↪related to domain \'PLUS3IT\' to local groups: 0x0
    06/23/2023 13:59:59:604 NetpManageLocalGroupsForJoin: INFO: No old domain groups to␣
↪process.
    06/23/2023 13:59:59:604 NetpJoinDomainLocal: Status of managing local groups: 0x0
    06/23/2023 13:59:59:604 NetpJoinDomainLocal: status of setting␣
↪ComputerNamePhysicalDnsDomain to \'plus3it.lab\': 0x0
    06/23/2023 13:59:59:604 NetpJoinDomainLocal: Controlling services and setting␣
↪service start type.
    06/23/2023 13:59:59:604 NetpJoinDomainLocal: Updating W32TimeConfig
    06/23/2023 13:59:59:620 NetpCompleteOfflineDomainJoin: status: 0x0
    06/23/2023 13:59:59:620 NetpJoinProvider2OLContinuePackagePartInstall: ignoring␣
↪Context=0 (work finished already).
    06/23/2023 13:59:59:620 NetpJoinProvider3OLContinuePackagePartInstall: ignoring␣
↪Context=0 (work finished already).
    06/23/2023 13:59:59:620 NetpProvContinueProvisioningPackageInstall: Provisioning␣
↪package installation completed successfully.
    06/23/2023 13:59:59:620 NetpProvContinueProvisioningPackageInstall: delete OP state␣
↪key status: 0x0.
    06/23/2023 13:59:59:620 NetpProvContinueProvisioningPackageInstall: status: 0xa99.
    06/27/2023 12:51:08:911 ----------------------------------------------------------------
↪----
    06/27/2023 12:51:08:911 NetpUnJoinDomain: unjoin from \'PLUS3IT\' using \'plus3it.
↪lab\\dom-joiner\' creds, options: 0x4
    06/27/2023 12:51:08:911         OS Version: 10.0
```

(continues on next page)

(continued from previous page)

```
    06/27/2023 12:51:08:911         Build number: 17763 (17763.rs5_release.180914-1434)
    06/27/2023 12:51:08:911         SKU: Windows Server 2019 Datacenter
    06/27/2023 12:51:08:911         Architecture: 64-bit (AMD64)
    06/27/2023 12:51:08:911 NetpUnJoinDomain: status of getting computer name: 0x0
    06/27/2023 12:51:08:911 NetpUnJoinDomain: DsrIsDeviceJoined returned false
    06/27/2023 12:51:08:911 NetpApplyJoinState: actions: 0x22b805a
    06/27/2023 12:51:08:927 NetpDsGetDcName: trying to find DC in domain \'PLUS3IT\',␣
↪flags: 0x1010
    06/27/2023 12:51:08:927 NetpDsGetDcName: found DC \'\\\\AWS-C2C4418931\' in the␣
↪specified domain
    06/27/2023 12:51:09:123 NetpApplyJoinState: status of connecting to dc \'\\\\AWS-
↪C2C4418931\': 0x0
    06/27/2023 12:51:10:141 NetpApplyJoinState: status of stopping and setting start␣
↪type of Netlogon to 16: 0x0
    06/27/2023 12:51:10:141 NetpApplyJoinState: NON FATAL: status of removing DNS␣
↪registrations: 0x0
    06/27/2023 12:51:10:141 NetpGetLsaMachineAccountInfoOld: status: 0x0
    06/27/2023 12:51:10:141 NetpApplyJoinState: status of getting LSA machine acct info␣
↪(old) 0x0
    06/27/2023 12:51:10:188 NetpManageMachineAccountWithSid: status of disabling account␣
↪\'IP-0A005598$\' on \'\\\\AWS-C2C4418931\': 0x0
    06/27/2023 12:51:10:188 NetpApplyJoinState: status of disabling account: 0x0
    06/27/2023 12:51:10:188 NetpApplyJoinState: status of setting LSA pri. domain: 0x0
    06/27/2023 12:51:10:188 NetpSetLsaMachineAccountInfoOld: status: 0x0
    06/27/2023 12:51:10:188 NetpApplyJoinState: status of setting LSA machine acct info␣
↪(old) 0x0
    06/27/2023 12:51:10:188 NetpApplyJoinState: status of clearing␣
↪ComputerNamePhysicalDnsDomain: 0x0
    06/27/2023 12:51:10:221 NetpApplyJoinState: status of removing from local groups: 0x0
    06/27/2023 12:51:10:259 NetpApplyJoinState: status of disconnecting from \'\\\\AWS-
↪C2C4418931\': 0x0
    06/27/2023 12:51:10:259 NetpUnJoinDomain: status: 0x0
    06/27/2023 12:51:10:274 ----------------------------------------------------------
↪----
    06/27/2023 12:51:10:274 NetpDoDomainJoin
    06/27/2023 12:51:10:274 NetpDoDomainJoin: using current computer names
    06/27/2023 12:51:10:274 NetpDoDomainJoin: NetpGetComputerNameEx(NetBios) returned 0x0
    06/27/2023 12:51:10:274 NetpMachineValidToJoin: \'IP-0A005598\'
    06/27/2023 12:51:10:274         OS Version: 10.0
    06/27/2023 12:51:10:274         Build number: 17763 (17763.rs5_release.180914-1434)
    06/27/2023 12:51:10:274         SKU: Windows Server 2019 Datacenter
    06/27/2023 12:51:10:274         Architecture: 64-bit (AMD64)
    06/27/2023 12:51:10:274 NetpMachineValidToJoin: status: 0x0
    06/27/2023 12:51:10:274 NetpJoinWorkgroup: joining computer \'IP-0A005598\' to␣
↪workgroup \'WORKGROUP\'
    06/27/2023 12:51:10:274 NetpValidateName: checking to see if \'WORKGROUP\' is valid␣
↪as type 2 name
    06/27/2023 12:51:16:366 NetpCheckNetBiosNameNotInUse for \'WORKGROUP\' [ Workgroup␣
↪as MACHINE]  returned 0x0
    06/27/2023 12:51:16:366 NetpValidateName: name \'WORKGROUP\' is valid for type 2
    06/27/2023 12:51:16:366 NetpJoinWorkgroup: status:  0x0
    06/27/2023 12:51:16:366 NetpDoDomainJoin: status: 0x0
```

(continues on next page)

(continued from previous page)

```
    06/27/2023 12:57:09:335 ----------------------------------------------------------------
↪----
    06/27/2023 12:57:09:335 NetpDoDomainJoin
    06/27/2023 12:57:09:335 NetpDoDomainJoin: using new computer names
    06/27/2023 12:57:09:335 NetpDoDomainJoin: NetpGetNewMachineName returned 0x0
    06/27/2023 12:57:09:335 NetpDoDomainJoin: NetpGetNewHostName returned 0x0
    06/27/2023 12:57:09:335 NetpMachineValidToJoin: \'IP-0A005598\'
    06/27/2023 12:57:09:335         OS Version: 10.0
    06/27/2023 12:57:09:335         Build number: 17763 (17763.rs5_release.180914-1434)
    06/27/2023 12:57:09:335         SKU: Windows Server 2019 Datacenter
    06/27/2023 12:57:09:335         Architecture: 64-bit (AMD64)
    06/27/2023 12:57:09:335 NetpMachineValidToJoin: status: 0x0
    06/27/2023 12:57:09:335 NetpJoinDomain
    06/27/2023 12:57:09:335         HostName: ip-0A005598
    06/27/2023 12:57:09:335         NetbiosName: IP-0A005598
    06/27/2023 12:57:09:335         Domain: plus3it.lab\\aws-c2c4418931.plus3it.lab
    06/27/2023 12:57:09:335         MachineAccountOU: (NULL)
    06/27/2023 12:57:09:335         Account: dom-joiner
    06/27/2023 12:57:09:335         Options: 0x403
    06/27/2023 12:57:09:335 NetpDisableIDNEncoding: no domain dns available - IDN␣
↪encoding will NOT be disabled
    06/27/2023 12:57:09:335 NetpJoinDomainOnDs: NetpDisableIDNEncoding returned: 0x0
    06/27/2023 12:57:09:398 NetUseAdd to \\\\aws-c2c4418931.plus3it.lab\\IPC$ returned␣
↪1326
    06/27/2023 12:57:09:398 NetpJoinDomainOnDs: status of connecting to dc \'\\\\aws-
↪c2c4418931.plus3it.lab\': 0x52e
    06/27/2023 12:57:09:398 NetpJoinDomainOnDs: Function exits with status of: 0x52e
    06/27/2023 12:57:09:398 NetpJoinDomainOnDs: NetpResetIDNEncoding on \'(null)\': 0x0
    06/27/2023 12:57:09:398 NetpDoDomainJoin: status: 0x52e
    06/27/2023 12:57:19:429 ----------------------------------------------------------------
↪----
    06/27/2023 12:57:19:429 NetpDoDomainJoin
    06/27/2023 12:57:19:429 NetpDoDomainJoin: using new computer names
    06/27/2023 12:57:19:429 NetpDoDomainJoin: NetpGetNewMachineName returned 0x0
    06/27/2023 12:57:19:429 NetpDoDomainJoin: NetpGetNewHostName returned 0x0
    06/27/2023 12:57:19:429 NetpMachineValidToJoin: \'IP-0A005598\'
    06/27/2023 12:57:19:429         OS Version: 10.0
    06/27/2023 12:57:19:429         Build number: 17763 (17763.rs5_release.180914-1434)
    06/27/2023 12:57:19:429         SKU: Windows Server 2019 Datacenter
    06/27/2023 12:57:19:429         Architecture: 64-bit (AMD64)
    06/27/2023 12:57:19:429 NetpMachineValidToJoin: status: 0x0
    06/27/2023 12:57:19:429 NetpJoinDomain
    06/27/2023 12:57:19:429         HostName: ip-0A005598
    06/27/2023 12:57:19:429         NetbiosName: IP-0A005598
    06/27/2023 12:57:19:429         Domain: plus3it.lab\\aws-c2c4418931.plus3it.lab
    06/27/2023 12:57:19:429         MachineAccountOU: (NULL)
    06/27/2023 12:57:19:429         Account: dom-joiner
    06/27/2023 12:57:19:429         Options: 0x403
    06/27/2023 12:57:19:429 NetpDisableIDNEncoding: no domain dns available - IDN␣
↪encoding will NOT be disabled
    06/27/2023 12:57:19:429 NetpJoinDomainOnDs: NetpDisableIDNEncoding returned: 0x0
    06/27/2023 12:57:19:476 NetUseAdd to \\\\aws-c2c4418931.plus3it.lab\\IPC$ returned␣
```

(continues on next page)

(continued from previous page)

```
↪1326
    06/27/2023 12:57:19:476 NetpJoinDomainOnDs: status of connecting to dc \'\\\\aws-
↪c2c4418931.plus3it.lab\': 0x52e
    06/27/2023 12:57:19:476 NetpJoinDomainOnDs: Function exits with status of: 0x52e
    06/27/2023 12:57:19:476 NetpJoinDomainOnDs: NetpResetIDNEncoding on \'(null)\': 0x0
    06/27/2023 12:57:19:476 NetpDoDomainJoin: status: 0x52e
    06/27/2023 12:57:39:510 ------------------------------------------------------------
↪----
    06/27/2023 12:57:39:510 NetpDoDomainJoin
    06/27/2023 12:57:39:510 NetpDoDomainJoin: using new computer names
    06/27/2023 12:57:39:510 NetpDoDomainJoin: NetpGetNewMachineName returned 0x0
    06/27/2023 12:57:39:510 NetpDoDomainJoin: NetpGetNewHostName returned 0x0
    06/27/2023 12:57:39:510 NetpMachineValidToJoin: \'IP-0A005598\'
    06/27/2023 12:57:39:510         OS Version: 10.0
    06/27/2023 12:57:39:510         Build number: 17763 (17763.rs5_release.180914-1434)
    06/27/2023 12:57:39:510         SKU: Windows Server 2019 Datacenter
    06/27/2023 12:57:39:510         Architecture: 64-bit (AMD64)
    06/27/2023 12:57:39:510 NetpMachineValidToJoin: status: 0x0
    06/27/2023 12:57:39:510 NetpJoinDomain
    06/27/2023 12:57:39:510         HostName: ip-0A005598
    06/27/2023 12:57:39:510         NetbiosName: IP-0A005598
    06/27/2023 12:57:39:510         Domain: plus3it.lab\\aws-c2c4418931.plus3it.lab
    06/27/2023 12:57:39:510         MachineAccountOU: (NULL)
    06/27/2023 12:57:39:510         Account: dom-joiner
    06/27/2023 12:57:39:510         Options: 0x403
    06/27/2023 12:57:39:510 NetpDisableIDNEncoding: no domain dns available - IDN␣
↪encoding will NOT be disabled
    06/27/2023 12:57:39:510 NetpJoinDomainOnDs: NetpDisableIDNEncoding returned: 0x0
    06/27/2023 12:57:39:574 NetUseAdd to \\\\aws-c2c4418931.plus3it.lab\\IPC$ returned␣
↪1326
    06/27/2023 12:57:39:574 NetpJoinDomainOnDs: status of connecting to dc \'\\\\aws-
↪c2c4418931.plus3it.lab\': 0x52e
    06/27/2023 12:57:39:574 NetpJoinDomainOnDs: Function exits with status of: 0x52e
    06/27/2023 12:57:39:574 NetpJoinDomainOnDs: NetpResetIDNEncoding on \'(null)\': 0x0
    06/27/2023 12:57:39:574 NetpDoDomainJoin: status: 0x52e
  ',
  'stderr': '
    Retry-TestCommand : Command [xAdd-Computer] failed
    At C:\\ProgramData\\Salt Project\\Salt\\var\\cache\\salt\\minion\\extfiles\\join-
↪domain\\JoinDomain.ps1:678 char:7
    +       Retry-TestCommand -Test xAdd-Computer -Args @{DomainName=$Domai ...
    +       ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        + CategoryInfo          : OperationStopped: (Command [xAdd-Computer]␣
↪failed:String) [Retry-TestCommand], RuntimeEx
       ception
        + FullyQualifiedErrorId : Command [xAdd-Computer] failed,Retry-TestCommand
  '
}
2023-06-27 12:57:39,859 [salt.state       :2458][INFO    ][5656] Completed state [& "C:\
↪ProgramData\Salt Project\Salt\var\cache\salt\minion\extfiles\join-domain\JoinDomain.ps1
↪" -DomainName "plus3it.lab" -TargetOU "" -UserName "dom-joiner" -Tries 3 -ErrorAction␣
↪Stop] at time 12:57:39.859161 (duration_in_ms=32050.5)
```

(continues on next page)

(continued from previous page)

```
[...elided...]
```

There can be other files in the `c:\watchmaker\logs\` directory, but the ones present will depend on what enterprise-integration features have been selected for `watchmaker` to attempt to execute and whether those integrations are configured to log independently.

There may be further log-files of interest, depending on how much execution-progress `watchmaker` has made and how `watchmaker` has been invoked. These will typically vary by build environment (e.g., when used with a CSP like Azure or AWS, on a physical server or a VM) and what tooling was used to invoke watchmaker.

The additional log-files of interest are typically generated by whatever Windows-specific userData payload-handler is leveraged. The known additonal log-files of interest will be enumerated in further sub-sections. If you are using watchmaker via userData payload and the handler is not enumerated below, please contribute to this project's documentation.

AWS:

When official Windows instances – ones published through the Amazon/Microsoft partnership – are launched into AWS and execute watchmaker via a userData payload, either of the following log files will be created:

- `C:\ProgramData\Amazon\EC2Launch\log\agent.log` (*see:* "EC2Launch" discussion-document)
- `C:\ProgramData\Amazon\EC2-Windows\Launch\Log\UserdataExecution.log` (*see:* "EC2Launch v2" discussion- document)

Which log file gets created will depend on the userData-handler used. Older versions of Windows Server (2012, 2016 and 2019) typically use the EC2 Launch handler. Newer versions of Windows Server (2022) use the EC2 v2 Launch-handler[1].

The `C:\ProgramData\Amazon\EC2-Windows\Launch\Log\UserdataExecution.log` Log-File

This file tracks the top-level execution of any tasks specified in a Windows Server EC2's userData payload. This file should *always* exist. The primary reasons that it may not exist are:

- The EC2 was launched from an AMI that leverages the `EC2Launch v2` method
- The EC2 was launched from an AMI that does not have the tooling to support parsing/executing a userData

Windows AMIs published through the Amazon/Microsoft partnership will always contain the tooling to support either the `EC2Launch` or `EC2Launch v2` parsing/execution of userData payloads:

- Windows Server 2022 and higher AMIs use the `EC2Launch v2` userData payload-handler
- Windows Server 2012, 2016 and 2019 AMIs use the `EC2Launch` userData payload-handler unless their AMI-names start with the string "`EC2LaunchV2-`"

To get a list of Windows AMIs that leverage the legacy `EC2Launch` userData payload-handler, use a (CLI) query similar to:

[1] Since the introduction of the EC2 v2 Launch-handler, official Windows Server AMIs for Server 2012, 2016 and 2019 have been being published. However, they are not the current default (as of the writing of this document). See the link to the EC2 v2 Launch discussion-document for details and caveats.

```
aws ec2 describe-images \
  --owner amazon \
  --filters 'Name=name,Values=Windows_Server-201*' \
  --query 'Images[].[CreationDate,ImageId,Name]' \
  --output text
```

The `C:\ProgramData\Amazon\EC2Launch\log\agent.log` Log-File

Warning: Watchmaker has not yet been tested with the EC2Launch v2 service. As of the writing of this document, the recommended userData content does not function under EC2Launch v2. This document is *very* beta, and primarily acts as a place-holder.

This file tracks the top-level execution of any tasks specified in a Windows Server EC2's userData payload. This file should *always* exist. The primary reasons that it may not exist are:

- The EC2 was launched from an AMI that leverages the (legacy) `EC2Launch` method
- The EC2 was launched from an AMI that does not have the tooling to support parsing/executing a userData

Windows AMIs published through the Amazon/Microsoft partnership will always contain the tooling to support either the `EC2Launch` or `EC2Launch v2` parsing/execution of userData payloads:

- Windows Server 2022 and higher AMIs use the `EC2Launch v2` userData payload-handler
- Windows Server 2012, 2016 and 2019 AMIs use the `EC2Launch` userData payload-handler unless their AMI-names start with the string "`EC2LaunchV2-`"

To get a list of Windows 2012, 2016 or 2019 AMIs that leverage the `EC2Launch v2` userData payload-handler, use a (CLI) query similar to:

```
aws ec2 describe-images \
  --owner amazon \
  --filters 'Name=name,Values=EC2LaunchV2-Windows_Server-201*' \
  --query 'Images[].[CreationDate,ImageId,Name]' \
  --output text
```

2.6 Hardening "Gotchas"

The hardening-content shipped with watchmaker includes some content that may result in degredation of the hardened-systems' user experience. We will try to document, here, those gotchas that we discover or that we are able to verify from user-submitted issue-reports.

2.6.1 Use of sudo broken after application of "extra" hardenings (EL7)

The STIG-handlers for each of RHEL-07-020020 and RHEL-07-020023 can break ability to `sudo` if applied.

Note:

1. These hardenings have, historically, not been universally applied to watchmaker-hardened systems. However, they are included with the hardening contents and some watchmaker-users do choose to execute them.
2. Use with more-recent spel AMIs may avoid the problems reported, as those AMIs already include updated SELinux-related sudoers configurations.

- RHEL-07-020020

  Rule Title: *The Red Hat Enterprise Linux operating system must prevent non-privileged users from executing privileged functions to include disabling, circumventing, or altering implemented security safeguards/countermeasures.*

  This handler for this STIG-ID makes sure that every local user with a `uid` and/or `gid` respectively greater than the `SYS_UID_MAX` and `SYS_GID_MAX` settings in the `/etc/login.defs` file has an SELinux user-confinement defined. These confinements can be viewed by executing:

  ```
  semanage login -l
  ```

  If a local user hasn't already had a user-confinement applied, this state will apply one. The state will search relevant Pillar-data to look for confinement-mappings and apply any that are defined. The relevant Pillar data is defined in the `/srv/watchmaker/salt/pillar/<ENVIRONMENT>/init.sls` file under the `ash-linux:lookup:sel_confine` map-object.

  One can check if watchmaker is aware of this map-object by typing:

  ```
  salt-call -c /opt/watchmaker/salt pillar.get ash-linux:lookup:sel_confine
  ```

  As the `root` user. If execution of this command returns only:

  ```
  local:
  ```

  Then the pillar-data is not present. If pillar-data *is* present, then it should return a list of SELinux user-confinements and each returned confinement will have a list of one or more usernames. For example:

  ```
  # salt-call -c /opt/watchmaker/salt pillar.get ash-linux:lookup:sel_confine
  local:
      |_
  ```

(continues on next page)

(continued from previous page)

```
      ----------
      staff_u:
          - ec2-user
    |_
      ----------
      unconfined_u:
          - ssm-user
```

Each username listed under a confinement will be given that confinement when RHEL-07-020020 is run.

- RHEL-07-020023

  Rule Title: *The Red Hat Enterprise Linux operating system must elevate the SELinux context when an administrator calls the sudo command.*

  This handler makes sure that every user of the `sudo` subsystem has an SELinux privilege-transition defined. These transitions aim to make it so that when users execute `sudo`, they don't have to pass any extra command-options to get the correct SELinux profile.

User Mapped As `user_u`

If the `ash-linux:lookup:sel_confine` map-object does not exist in the Pillar-data, *every* local user without an existing confinement will get mapped to the `user_u` confinement. This confinement-level will significantly restrict the things that the user-account can do, inclusive of using `sudo`. In the case of `sudo` these restrictions will manifest similarly to:

```
$ sudo -i
sudo: PERM_SUDOERS: setresuid(-1, 1, -1): Operation not permitted
sudo: no valid sudoers sources found, quitting
sudo: setresuid() [0, 0, 0] -> [1004, -1, -1]: Operation not permitted
sudo: unable to initialize policy plugin
$
```

User Mapped As `staff_u`

If the previously-discussed pillar-data declares that a given username should be mapped to the `staff_u` user-confinement, execution of `sudo` *may* result in an error. A quick `sudo -i` will show:

```
$ sudo -i
-bash: /root/.bash_profile: Permission denied
```

Some further actions might show output similar to the following:

```
-bash-4.2# id -Z
staff_u:staff_r:staff_t:s0-s0:c0.c1023

-bash-4.2# ls -al
ls: cannot access .bashrc: Permission denied
ls: cannot access .bash_history: Permission denied
ls: cannot access .tcshrc: Permission denied
ls: cannot access .bash_profile: Permission denied
ls: cannot access .bash_logout: Permission denied
```

(continues on next page)

(continued from previous page)

```
ls: cannot access install.sh: Permission denied
ls: cannot access .cshrc: Permission denied
total 24
dr-xr-x---.  6 root root 4096 Aug 17 14:35 .
dr-xr-xr-x. 19 root root 4096 Aug 17 12:51 ..
-??????????  ? ?    ?       ?            ? .bash_history
-??????????  ? ?    ?       ?            ? .bash_logout
-??????????  ? ?    ?       ?            ? .bash_profile
-??????????  ? ?    ?       ?            ? .bashrc
-??????????  ? ?    ?       ?            ? .cshrc
drwx------.  2 root root 4096 Aug 17 13:22 .ssh
-??????????  ? ?    ?       ?            ? .tcshrc
```

The errors are due to the sudo action selecting the default `staff_u:staff_r:staff_t:s0-s0:c0.c1023` SELinux rights-mapping.

Workaround:

To work around, invoke sudo as follows:

```
$ sudo -i -r sysadm_r -t sysadm_t
```

The user should receive no SELinux errors before the root prompt is displayed, nor should they receive errors from operations like `ls ${HOME}`. Similarly, if they execute `id -Z`, they should get output similar to:

```
# id -Z
staff_u:sysadm_r:sysadm_t:s0-s0:c0.c1023
```

User Mapped As `unconfined_u`

There are a few ways that users get assigned this confinement: the user was explicitly created with that confinement assigned; they login using a third-party authentication-service like Active directory; or the previously-mentioned pillar-data was configured to map them to that confinement. Typically, so-mapped users will not experience any SELinux-related permissions problems. However, if an SELinux role-transition has been defined in the `sudoers` subsystem (as is done when RHEL-07-020023 is run), the user may experience an error like:

```
$ sudo -i
sudo: unconfined_u:sysadm_r:sysadm_t:s0-s0:c0.c1023 is not a valid context
```

Workaround:

To work around, invoke sudo as follows:

```
$ sudo -i -r unconfined_r -t unconfined_t
```

2.6.2 X11-Forwarding Via SSH is "Broken" (EL8)

The STIG-handlers for:

- RHEL-08-040340 (and its Oracle equivalent, OL08-00-040340):

  RHEL 8 remote X connections for interactive users must be disabled unless to fulfill documented and validated mission requirements.

- RHEL-08-020041 (and its Oracle equivalent, OL08-00-020041):

  RHEL 8 must ensure session control is automatically started at shell initialization

These settings will negatively-impact expected behaviors around X11-forwarding through SSH tunnels after application.

Symptoms

An error like "Can't open display" is emitted when attempting to launch X11 client-applications

Things to Verify

- Is `X11Forwarding` disabled in the `/etc/ssh/sshd_config` file
- Is the `xauth` utility available: this utility is installed at `/usr/bin/xauth` through the installation of the `xorg-x11-xauth` RPM
- Was X11-forwarding requested by the SSH client when establishing the connection to the remote host

Fixes

`X11Forwarding` is disabled in the `/etc/ssh/sshd_config` file

Do the following to allow the use of X11-forwarding over SSH:

1. Execute `sudo grep -P '^(\s*)X11' /etc/ssh/sshd_config`. If the previous command returns null or shows a value of `no`
2. Update the `/etc/ssh/sshd_config` file to:
   - Ensure that any (uncommented) value for `X11Forwarding` is set to `yes`
   - Add a `X11Forwarding yes` line to the file if no uncommented `X11Forwarding` lines exist
3. Restart the `sshd` service

Note: The preceding will result in a scan-finding on any system that it has been executed on. The "…unless to fulfill documented and validated mission requirements" component of the STIG-rule means that provision of a documented reason for the enablement should allow the finding to be dismissed.

No `xauth` utility available

- Ensure that the `xorg-x11-xauth` RPM is installed
- Ensure that `/usr/bin` is in the user's `PATH` environment

Ensure that X11 forwarding has been requested by your SSH client

The methods for requesting X11 forwarding are specific to each SSH client. OpenSSH clients typically require including the `-Y` flag when requesting a connection. Consult vendor-documents for the proper setup of other SSH clients.

Next Steps

Assuming that all of the above verifications and associated fixes have been done and things still are not working as expected, it's likely that, when the login processes start up the `tmux` service, that the (tunneled) `DISPLAY` environment variable has not been properly set. This may occur because, when the `tmux` service is activated, the `DISPLAY` variable was not propagated from the initial login-shell to the `tmux`-managed subshell(s). If all other necessary components for setting up X-over-SSH are in place, the following should allow the session-user to set up an appropriate `DISPLAY` value within their `tmux` session-window:

1. Verify that your `${HOME}/.Xauthority` file exists
2. List out the authorization entries in the authority-file (execute `xauth list`). This should result in output like:

   ```
   x-test.wam.lab/unix:10  MIT-MAGIC-COOKIE-1  dbb1c620b838faf7d5e5717d4a217a7c
   ```

   On a freshly-launched system, there should be one and only one entry. If there's more than one entry, it means that either the file is stale or that more than one login-session has been concurrently opened to the remote host
3. Export the environment variable `DISPLAY`, setting it to `localhost:<NUMBER>`. The value of `<NUMBER>` will be the digit after the `<hostname>/unix:` string in the `xauth list` output. Typically, this will mean executing `export DISPLAY=localhost:10`.

Once the above has been done, attempts to launch X11 clients *should* result in them displaying to the display of the system the operator has originally SSHed from.

Note: If one takes advantage of `tmux`'s ability to multiplex terminal and one wishes to be able to launch X11 apps from any `tmux`-managed session-window, it will be necessary to export the `DISPLAY` variable in *each* `tmux` session-window.

2.6.3 OpenSSH RSAv2 Keys Don't Work (EL8)

Background

The OpenSSH Daemon shipped with the most-recent versions of RHEL 8 (and derivatives), implements the deprecation of SHA1-signed SSH keys for key-based authentication that's now part of OpenSSH 8.8 and higher. As such, any SSH keys used for key-based authentication will need to be signed using a SHA2 algorithm (SHA-256 or SHA-512).

Workarounds

For users of self-managed keys, this means that one needs to present an SHA-256 or SHA-512 signed OpenSSH key when using RSAv2 keys for key-based logins. Such keys can be generated in a couple ways:

- Use either `rsa-sha2-256` or `rsa-sha2-512` when using `ssh-keygen`'s `-t` option for generating a new key
- Use `ssh-keygen` on a FIPS-enabled, EL8+ operating system
- Use a CSP's key-generation tool (AWS's commercial region's EC2 key-generation capability is known to create conformant RSAv2 keys)

For users of organizationally-issued SSH keys - be they bare files or as delivered via a centrally-managed SmartCard (such as a PIV or CAC) or other token - it will be necessary for the key-user to work with their organization to ensure that updated, conformant keys are issued.

Symptoms

Depending on the SSH client, the key may silently fail to work or it may print an error. If an error is printed, it will usually be something like:

```
Load key "/path/to/key-file": error in libcrypto
```

With or without the printing of the error, the key will be disqualified and the server will request the client move on to the next-available authentication-metho (usually password).

If one is able to use other means to access a system and view its logs, one will usually find errors similar to:

```
Feb 09 12:10:50 ip-0a00dc73 sshd[2939]: input_userauth_request: invalid user ec2-user␣
↪[preauth]
```

Or

```
Feb 09 12:10:50 ip-0a00dc73 sshd[2939]: input_userauth_pubkey: key type ssh-rsa not in␣
↪PubkeyAcceptedKeyTypes [preauth]
```

In the `/var/log/secure` logs.

Note: The deprecated SHA-1 issuse is not a watchmaker issue. It is generically applicable to Red Hat's OpenSSH version on EL8-bsed systems. However, because most people will encounter the issue after having run watchmaker, we opted to include it in this project's "Gotchas" documentation for the benefit of watchmaker-users that might come here for answers

2.7 Frequently Asked Questions

2.7.1 How do I know if watchmaker has installed?

To determine whether watchmaker is installed, the simplest method is to run the command `watchmaker --help`. If it displays the cli help page, watchmaker is installed. Another option is to check `pip list | grep watchmaker`.

2.7.2 What do I do if watchmaker failed to install?

First, review the *installation* document. Then double-check the output of a failed installation. Usually, the output points pretty clearly at the source of the problem. Watchmaker can be re-installed over itself with no problem, so once the root cause is resolved, simply re-install watchmaker.

2.7.3 Why does the watchmaker install fail if my system is FIPS enabled?

This is primarily a question for Red Hat (and derived distributions). As of this writing, the `pip` utility in all Red Hat releases up through 7.4.1708, default to looking for pypi packages signed with MD5 signatures. If you've enabled FIPS (or are using a build that has FIPS pre-enabled), MD5 is disabled in the kernel (due to being a weak hashing-method). You can either disable FIPS (not recommended) or explicitly force `pip` to use a different signature-index. The latter is detailed in the Linux section of the *usage* document.

2.7.4 How do I know if watchmaker has completed without errors?

By default, watchmaker will reboot the system after a sucessful execution. Therefore, if the system reboots, watchmaker executed successfully. If you are investigating sometime after watchmaker completed, check the logs for errors. If anything fails, watchmaker will suppress the reboot. (Though note that the `--no-reboot` flag can be used to suppress the reboot even after a successful execution.)

You can also test the watchmaker exit code programmatically. If watchmaker fails, it will return a non-zero exit code. If watchmaker completes successfully, it will return an exit code of zero. You would typically pass the `--no-reboot` flag if you intend to test the exit code and determine what to do from there.

2.7.5 What do I do if watchmaker failed to complete or completes with errors?

Start by checking the logs generated by watchmaker. The logs are stored in the directory specified by the `--log-dir` argument. Search the log for entries that have `[ERROR]`, this will give you a starting point to begin troubleshooting. Also, if a salt state failed, look for the pattern `Result:  False`. If it is not an obvious or simple issue, feel free to create an issue on the watchmaker github page. If there is a salt_call.debug.log in the watchmaker log directory, you can look for `[ERROR]` messages in there as well. However, this log file can be very noisy and a message with the error label may not be related to the error you are encountering.

2.7.6 Does watchmaker support Enterprise Linux 7?

Watchmaker is supported on RedHat 7 and CentOS 7. See the *index* page for a list of all supported operating systems.

2.7.7 Does watchmaker support Enterprise Linux 8?

Watchmaker is supported on RedHat 8, CentOS 8 Stream, and Oracle Linux 8. See the *index* page for a list of all supported operating systems.

2.7.8 How can I exclude salt states when executing watchmaker?

The Salt worker in Watchmaker supports an `exclude_states` argument. When present, the value is passed directly to the `exclude` option of the salt highstate execution module. To use this option with watchmaker from the command line, pass the argument `--exclude-states <sls_glob>`. For example:

```
# Exclude the state "foo" with an exact match
watchmaker --exclude-states foo

# Exclude all state names that begin with "foo"
watchmaker --exclude-states foo*

# Exclude multiple states "foo" and "bar" with an exact match
watchmaker --exclude-states foo,bar
```

2.7.9 Can I use the underlying salt functionality directly?

Yes, by passing watchmaker's salt configuration directory to the salt command, using the `-c|--config-dir` argument:

- Linux: `/opt/watchmaker/salt`
- Windows: `C:\Watchmaker\salt\conf`

For example:

```
# -c|--config-dir
salt-call -c /opt/watchmaker/salt state.show_top
```

2.7.10 Can I use watchmaker to toggle my RedHat/Centos host's FIPS mode?

Yes, indirectly. Because watchmaker implements most of its functionality via SaltStack modules, you can directly-use the underlying SaltStack functionality to effect the desired change. This is done from the commandline - as root - by executing:

- Disable FIPS-mode: `salt-call -c /opt/watchmaker/salt ash.fips_disable`
- Enable FIPS-mode: `salt-call -c /opt/watchmaker/salt ash.fips_enable`

And then rebooting the system.

2.7.11 How do I install watchmaker when I am using Python 2.6?

While Watchmaker no longer offically supports Python 2.6, you may use the last version where it was tested, Watchmaker 0.21.7. That version includes pins on dependencies that will work for Python 2.6.

However, there are three python "setup" packages needed just to install `watchmaker`, and these packages cannot be platform-restricted within the `watchmaker` package specification.

Below is the list of packages in question, and the versions that no longer support Python 2.6:

- `pip>=10`
- `wheel>=0.30.0`
- `setuptools>=37`

In order to install pip in Python 2.6, you can get it from:

- https://bootstrap.pypa.io/pip/2.6/get-pip.py

Once a Python 2.6-compatible `pip` version is installed, you can install compatible versions of the other packages like this:

```
python -m pip install --upgrade "pip<10" "wheel<0.30.0" "setuptools<37"
```

You can then *install watchmaker* by restricting the watchmaker version to the last version tested with Python 2.6:

```
python -m pip install "watchmaker==0.21.7"
```

2.7.12 How do I get Watchmaker release/project notifications?

Users may use an RSS reader of their choice to subscribe to the Watchmaker Release feed to get notifications on Watchmaker releases. The Watchmaker RSS release feed is https://github.com/plus3it/watchmaker/releases.atom.

Users can also "watch" the GitHub project to receive notifications on all project activity, https://github.com/plus3it/watchmaker/subscription.

2.8 Supported SCAP Benchmarks

2.8.1 Windows

- Microsoft Windows Server STIG Benchmark (2019)
- Microsoft Windows Server STIG Benchmark (2016)
- Microsoft Windows Server STIG Benchmark (2012-r2)
- Microsoft Windows STIG Benchmark (10)
- Microsoft .NET Framework 4 Benchmark
- Internet Explorer STIG Benchmark

2.8.2 Linux

- Red Hat Enterprise Linux STIG Benchmark (EL7 and EL8)
- OpenSCAP Security Guide (EL7 and EL8)

  New benchmark versions are incorporated as they are released

2.9 Common Scan Findings

There is frequently more than one way to achieve a given hardening-recommendation. As such, generic security scanners may produce alerts/findings that are at odds with the actual system state implemented by Watchmaker. The following are frequently-cited findings and explanations for why a scanner may alert on the Watchmaker-managed configuration-state.

2.9.1 Common Scan Findings for EL7

Findings Summary-Table

Finding Summary	Finding Identifiers
Use Only FIPS 140-2 Validated Ciphers	SV-86845 RHEL-07-040110
Use Only FIPS 140-2 Validated MACs	SV-86877 RHEL-07-040400
Modify the System Login Banner	SV-86487 RHEL-07-010050
Enable Smart Card Login	SV-86589 RHEL-07-010500
Configure the Firewalld Ports	SV-86843 RHEL-07-040100
Set Default firewalld Zone for Incoming Packets	SV-86939 RHEL-07-040810
Disable Kernel Parameter for IP Forwarding	SV-86933 RHEL-07-040740
The Installed Operating System Is Vendor Supported	SV-86621 RHEL-07-020250
Install McAfee Virus Scanning Software	SV-86837 RHEL-07-032000
Enable FIPS Mode in GRUB2	SV-86691 RHEL-07-021350
Configure AIDE to Use FIPS 140-2 for Validating Hashes	SV-86697 RHEL-07-021620
Verify and Correct Ownership with RPM	SV-86473 RHEL-07-010010
Verify and Correct File Permissions with RPM	SV-86473 RHEL-07-010010
Ensure Users Re-Authenticate for Privilege Escalation - sudo NOPASSWD	SV-86571 RHEL-07-010340
Operating system must display the date and time of the last successful account logon upon logon	SV-86899 RHEL-07-040530
Operating system must be configured so that the audit system takes appropriate action when the audit storage volume is full	SV-86711 RHEL-07-030320
Operating system must be configured to off-load audit logs onto a different system or storage media from the system being audited	SV-95729 RHEL-07-030201
User Must Not Be Allowed To Change Password More-frequently than once per 24 hours	SV-86551 RHEL-07-010240
User Must Change Password At Least Once Every Sixty Days	SV-86555 RHEL-07-010260
User Must Be Provided Adequate Warning Of Password-Expiration	
User Account Must Be Expired N Days After Password Has Expired	SV-86565 RHEL-07-010310
For Operating Systems Using DNS Resolution, At Least Two Name Servers Must Be Configured	SV-204608 RHEL-07-040600
The OS Must Elevate The SELinux Context When An Administrator Calls The Sudo Command	SV-250314 RHEL-07-020023

Use Only FIPS 140-2 Validated Ciphers

Invalid Finding:

Watchmaker implements setting valid through EL7 STIGv2R6 (released: October 2019)

Use Only FIPS 140-2 Validated MACs

Invalid Finding: Watchmaker implements setting valid through EL7 STIGv2R6 (released: October 2019)

Modify the System Login Banner

Invalid Finding:

Watchmaker implements site-prescribed banner. Scan-profile's regex may not be flexible enough to match the site-prescribed banner as implemented by watchmaker.

Enable Smart Card Login

Conditionally-Valid Finding:

Smart Card Login use and configuration is site-specific. Site has not provided specification for implementing this setting within scanned context.

Configure the Firewalld Ports

Invalid Finding:

Watchmaker implements setting. However, scanner regex may not be sufficiently-flexible in its specification.

Set Default firewalld Zone for Incoming Packets

Conditionally-Valid Finding:

Enabling "drop" as the default firewald zone breaks things like ping-sweeps (used by some IPAM solutions, security-scanners, etc.). Some sites will request the "drop" zone not be used. Scan-profiles should be updated to reflect the need to not have "drop" be the active zone.

Disable Kernel Parameter for IP Forwarding

Invalid Finding:

The prescribed `net.ipv4.ip_forward` value is set by watchmaker in `/etc/sysctl.d/99-sysctl.conf`. Executing `sysctl net.ipv4.ip_forward` on watchmaker-hardened system returns expected `net.ipv4.ip_forward = 0` result

The Installed Operating System Is Vendor Supported

Invalid Finding:

No programmatic validation or remediation prescribed or universally-implementable: requires manual validation with OS-vendor lifecycle information page(s).

Install McAfee Virus Scanning Software

Conditionally-Valid Finding:

- Where configured to do so, watchmaker will install HBSS or VSEL. Any scan-findings on systems watchmaker has been configured to install HBSS or VSEL are typically due to version mismatches between installed and scanned-for versions
- Where required/scanned for but not installed, site will need to specify automatable installation-method that will produce match againste scanned-for configuration
- Where not required, scanner should either be reconfigured not to scan for presence or scan-results should be ignored

Enable FIPS Mode in GRUB2

Conditionally-Valid Finding:

Both spel and watchmaker implement `fips=1` by default. If finding occurs, either:

- There is an error in scanner's validation-method
- System has been intentionally de-configured for FIPS — typically due to hosted-software's requirements — and scanned-system will need to be granted a deployment security-exception.

Configure AIDE to Use FIPS 140-2 for Validating Hashes

Invalid Finding:

Because there is more than one way to implement this setting, scanners typically do not perform a real scan for this setting. Instead some scanners implement a null-test to flag the configuration-item to try to force a manual review. Watchmaker implements this configuration-titem by setting `NORMAL = FIPSR+sha512` in the `/etc/aide.conf` file: may be manually validated by executing `grep NORMAL\ = /etc/aide.conf`.

Verify and Correct Ownership with RPM

Invalid Finding:

- Flags on system-journal ownership: Journal ownership settings are automatically reset by systemd (upon reboot) after hardening has run. Currently, no means of permanently remediating is possible.
- Similarly, if HBSS or VSEL is installed, scan may flag on user-ownership depending on how site specifies installation of HBSS or VSEL. One would reasonably expect similar for other, third-party packages. "Fixing" (per STIG guidance) would likely break the functioning of the HBSS/VSEL (or third-party) software

Verify and Correct File Permissions with RPM

Invalid Finding:

- Flags on system-journal ownership: Journal ownership settings are automatically reset by systemd (upon reboot) after hardening has run. Currently, no means of permanently remediating is possible.
- May also flag on vendor-delivered CA-trust files which are dynamicly-injected into relevant trust-stores. Currently, no known means of permanently remediating is possible.
- May flag on third-party tools' (e.g., Splunk) config, log and other files

Ensure Users Re-Authenticate for Privilege Escalation - sudo NOPASSWD

Conditionally-Valid Finding:

Flagged-configuration is frequently required for properly enabling a "break-glass" account at provisioning-time. This is especially so in consoleless environments (like AWS). Disable scan or ignore scan-findings when such accounts are required.

Operating system must display the date and time of the last successful account logon upon logon

Invalid Finding:

Some scanners implement a scan equivalent to:

```
grep -P '^[\s]*[^\s#]+[ \t]+[\[\]\w=]+[ \t]+pam_lastlog\.so[ \t]+([\S \t]+)\s*$' /etc/
↪pam.d/postlogin
```

To try to determine if PAM's `showfailed` module is properly activated. These scanners typically only expect a single line of output that looks like:

```
session    required       pam_lastlog.so showfailed
```

However, on a system that watchmaker has been applied to, the scan-return will typically look like:

```
session    required       pam_lastlog.so showfailed
session    [default=1]    pam_lastlog.so nowtmp showfailed
session    optional       pam_lastlog.so silent noupdate showfailed
```

If the scanner does not properly handle this multi-line output, it will report a failure even though the required configuration-fixes are actually in place and functioning as desired.

Operating system must be configured so that the audit system takes appropriate action when the audit storage volume is full

Invalid Finding:

The `disk_full_action` *is* configured. However, it is not configured where scanners may be configured to look for it. The STIG-prescribed method expects configuration through the `audisp-remote` subsystem. Since configuration of the `audisp-remote` subsystem is inherently site-specific, generic executions of watchmaker do not attempt to configure it. Instead, watchmaker handles the `disk_full_action` configuration-item via the *main* audit subsystem. This can be confirmed by executing:

```
( find /etc/audisp -type f ; find /etc/audit -type f ) | xargs grep disk_full_action
```

Executing the above *should* return something like:

```
/etc/audit/auditd.conf:disk_full_action = SUSPEND
```

Operating system must be configured to off-load audit logs onto a different system or storage media from the system being audited

Invalid Finding:

Configuration of the `audisp-remote` subsystem is inherently site-specific: quite frequently, the `audisp-remote` subsystem is wholly supplanted by other offload-methods (e.g., Splunk, FluentBit, CloudWatch Logs, etc.). Therefore, neither generic executions of watchmaker nor executions that include configuration of `audisp-remote` alternatives will attempt to configure it.

User Must Not Be Allowed To Change Password More-Frequently than once per 24 hours

Typically caused when a user is created via a service/process like `cloud-init`: the resulting user may not have its password-aging `mindays` parameter (field #4 in `/etc/shadow`) set

User Must Change Password At Least Once Every Sixty Days

Typically caused when a user is created via a service/process like `cloud-init`: the resulting user may not have its password-aging `maxdays` parameter (field #5 in `/etc/shadow`) set

User Must Be Provided Adequate Warning Of Password-Expiration

Typically caused when a user is created via a service/process like `cloud-init`: the resulting user may not have its password-aging `warndays` parameter (field #6 in `/etc/shadow`) set

User Account Must Be Expired N Days After Password Has Expired

Typically caused when a user is created via a service/process like `cloud-init`: the resulting user may not have its password-aging `inactivedays` parameter (field #7 in `/etc/shadow`) set

For Operating Systems Using DNS Resolution, At Least Two Name Servers Must Be Configured

Conditionally Valid:

Only valid in environments where individually-defined DNS servers are not highly-available.

When deployed into environments where DNS is provided through a highly-available service with a highly-available service-name, only one DNS server will be configured into the host's `/etc/resolv.conf` – typically by way of a DHCP option-set.

The OS Must Elevate The SELinux Context When An Administrator Calls The Sudo Command

Conditionally Valid:

Implementation of this finding's technical controls changes how the `sudo` commands are executed. Some EL7 tooling (at least one third-party authentication subsystem is known to break under this new control) is incompatible with implementing this control. For systems where this control breaks functionality, and must be disabled, this will be a valid finding that should be included in any exception documentation and associated organizational-processes. Otherwise the system should be configured to meet this control.

Further Notes:

1. Implementing this control can have significant user-education requirements and can also adversely-impact legacy automation. While these *should* be non-fatal problems – only requiring user-education or fine-tuning of legacy automation, the control still should be implemented.
2. As implemented in this project, the modifications to the relevnt `/etc/sudoers.d` files may create sub-optimal SELinux transistions. If so, it will be up to the watchmaker-user to deactivate the `ash-linux.el7.STIGbyID.cat2.RHEL-07-020023` (see the pillar.example file in the ash-linux-formula project; see also the associated README file for further elaboration) and then provide their own mapping-modifications as a substitute. Deactivation can be done via the `ash-linux:lookup:skip-stigs` list-variable in Pillar.

2.9.2 Common Scan Findings for EL8

Findings Summary-Table

A few scans performed against EL8 systems are version-dependent. Watchmaker is designed to ensure that a given EL8 host is running at the latest-available EL8 minor-release version. Some of the version-dependent scans are for versions (well) prior "the latest-available EL8 minor-release version". The person responding to scan-findings should make sure to notice if the findings-text includes mention of specific EL8 minor-release version or version-ranges and compare that to the EL8 minor-release of the scanned system. If the version/version-range is less than that of the scanned version, the scan result may be immediately flagged as "**INVALID FINDING**". Anything that cannot be immediate flagged in this way should be checked against the following table of known findings[1].

[1] Do not try to perform an exact-match from the scan-report to this table. The findings table's link-titles are distillations of the scan-findings title-text rather than being verbatim copies.

Finding Summary	Finding Identifiers
Prevent System Daemons From Using Kerberos For Authentication	V-230238 RHEL-08-010161
Users Must Provide A Password For Privilege Escalation	V-230271 RHEL-08-010380
A Separate Filesystem Must Be Used For the /tmp Directory	V-230295 RHEL-08-010543
The OS must mount /tmp with the nodev option	V-230511 RHEL-08-040123
The OS must mount /tmp with the nosuid option	V-230512 RHEL-08-040124
The OS must mount /tmp with the noexec option	V-230514 RHEL-08-040125
The OS Must Ensure Session Control Is Automatically Started At Shell Initialization	V-230349 RHEL-08-020041
User Account Passwords Must Be Restricted To A 60-Day Maximum Lifetime	V-230367 RHEL-08-020210
OS Must Prohibit Password Reuse For A Minimum Of Five Generations	V-230368 RHEL-08-020220
The Installed Operating System Is Not Vendor Supported	V-230221 RHEL-08-010000
All remote access methods must be monitored	V-230228 RHEL-08-010070
All Content In A User's Home Directory Must Be Group-Owned By The Primary User	V-244532 RHEL-08-010741
"Only Authorized Local User Accounts Exist on Operating System" is always flagged	V-230379 RHEL-08-020320
All Interactive User Home Directory Files Must Be Mode 0750 Or Less Permissive	V-244531 RHEL-08-010731
Add nosuid Option to /boot	V-230300 RHEL-08-010571
Configure Multiple DNS Servers in /etc/resolv.conf	V-230316 RHEL-08-010680
Enable Certmap in SSSD	V-230355 RHEL-08-020090
Verify that Shared Library Directories Have Root Ownership	V-251709 RHEL-08-010351
Oracle Linux 8 STIGs Specify Conflicting ClientAliveCountMax values	V-248552 OL08-00-010200
Record Events When Privileged Executables Are Run	V-248722 OL08-00-030000
EL 8 systems less than v8.4 must configure the password complexity module in the system-auth allow three retries or less	V-251714 RHEL-08-020102
EL 8 must enable the hardware random number generator entropy gatherer service	V-230285 RHEL-08-010471

Prevent System Daemons From Using Kerberos For Authentication

Condtionally-valid Finding:

If an EL8 system is bound to Active Directory – or other Kerberos-enabled centralized authentication-source – or is *acting as* a Kerberos domain controller (KDC), the presence of an `/etc/krb5.keytab` file is mandatory.

If the scanned system does not have the `krb5-workstation` or `krb5-server` packages installed and *any* `.keytab` files are found in the `/etc` directory, this is a valid finding.

Users Must Provide A Password For Privilege Escalation

Condtionally-valid Finding:

If a `sudo`-enabled user is password-enabled, they should be prompted for that password in order to escalate privileges.

If a `sudo`-enabled user is *not* password-enabled (e.g, if the user is used only with key- or token-based authentication), they should not be prompted for any password. Forcing use of passwords for `sudo`-enabled users that do not have passwords will render those user-accounts unable to escalate privileges. Failure to exempt such users will render "break glass" and similar accounts not usable for their intended function.

A Separate Filesystem Must Be Used For the `/tmp` Directory

Invalid Finding:

When using Amazon Machine Images, Azure VM-templates, or the like, that have been built using the spel automation, the `/tmp` directory in the resultant EC2 (VM, etc.) is hosted on a psuedo-filesystem of type `tmpfs`. This is done using the `tmp.mount` systemd unit. Many security-scanning tools that look for `/tmp`-related mount-information do not know how to properly scan when `tmp` is used this way and will, as a result, report a (spurious) finding.

Proof of Correctness:

To validate that the required-setting is *actually* present, execute:

```
grep -w /tmp /proc/mounts
```

If this returns null, the scan-result is valid; otherwise the scan-result is *invalid*.

The OS must mount `/tmp` with the `nodev` option

Invalid Finding:

When using Amazon Machine Images, Azure VM-templates, or the like, that have been built using the spel automation, the `tmp.mount` systemd unit is used to manage mounting of the `tmpfs`-based `/tmp` directory. Mount options – such as `nodev` – are defined through two files:

- `/usr/lib/systemd/system/tmp.mount`: This file contains the vendor-defined defaults and is installed via the `systemd` RPM
- `/etc/systemd/system/tmp.mount.d/options.conf`: This file is installed via watchmaker's state-handler, `ash-linux.el8.STIGbyID.cat2.RHEL-08-040123`. This file overrides the values held in the vendor-managed `systemd` RPM's file

Many security-scanners do not know how to find the mount-options for the `/tmp` (pseudo) filesystem when it is managed via systemd and uses these files to set the mount options. As a result, such scanners will report a (spurious) finding

Proof of Correctness:

To validate that the required-setting is *actually* present, execute:

```
grep -w /tmp /proc/mounts | grep nodev
```

If this returns null, the scan-result is valid; otherwise the scan-result is *invalid*.

The OS must mount `/tmp` with the `nosuid` option

Invalid Finding:

As with the "The OS must mount `/tmp` with the nodev option" finding, this finding is due to an incompatibility between how the scanner checks for the setting and how the setting is actually implemented.

Proof of Correctness:

To validate that the required-setting is *actually* present, execute:

```
grep -w /tmp /proc/mounts | grep nosuid
```

If this returns null, the scan-result is valid; otherwise the scan-result is *invalid*.

The OS must mount `/tmp` with the `noexec` option

Invalid Finding:

As with the "The OS must mount `/tmp` with the nodev option" finding, this finding is due to an incompatibility between how the scanner checks for the setting and how the setting is actually implemented.

Proof of Correctness:

To validate that the required-setting is *actually* present, execute:

```
grep -w /tmp /proc/mounts | grep noexec
```

If this returns null, the scan-result is valid; otherwise the scan-result is *invalid*.

The OS Must Ensure Session Control Is Automatically Started At Shell Initialization

Invalid Finding:

As implemented, watchmaker places an `/etc/profile.d/tmux.sh` file that looks like:

```
# Check if shell is interactive
if [[ $- == *i* ]] && [[ $( rpm --quiet -q tmux )$? -eq 0 ]]
then
   parent=$( ps -o ppid= -p $$ )
   name=$( ps -o comm= -p $parent )

   # Check if controlling-process is target-value
   case "$name" in
      sshd|login)
         exec tmux
         ;;
   esac
fi
```

This file addresses the concerns of the STIG finding-ID, but does so in a functionally-safer way. The additional 'safing' included in the watchmaker-placed script may cause scanners that are too-inflexibly coded to spuriously declare a finding.

User Account Passwords Must Be Restricted To A 60-Day Maximum Lifetime

Invalid Finding:

Some, locally-managed user's accounts are configured *only* for token-based logins (SSH keys, GSSAPI, etc.). The accounts are typically configured with no passwords. Some of these accounts also serve a "break-glass" function. If passwordless accounts are configured with password-expiry enabled, they may become no longer fit for purpose once they've reached their expiry.

Many scanners are not adequately configured to differentiate between passwordless and password-enabled locally-managed accounts. Typically, poorly-configured scanners will execute a compliance-test equivalent to:

```
awk -F: '$5 > 60 { print $1 " " $5 }' /etc/shadow
awk -F: '$5 <= 0 { print $1 " " $5 }' /etc/shadow
```

Or, expressed more compactly:

```
awk -F: '$5 > 60 || $5 <= 0 { print $0 }' /etc/shadow
```

If so, such scanners will assert a finding that is not actually valid for locked-password accounts.

Proof of Correctness:

To validate that passwordless accounts are properly configured, instead execute:

```
awk -F: '$2 !~ /^[!*]*/ && ( $5 > 60 || $5 <= 0 ) { print $0 }' /etc/shadow
```

The above adds the further check of each line of the `/etc/shadow` file's second field (hashed password string) for the tokens indicating a locked-password account (`!` and/or `*`). Adding this further check should yield a null return

OS Must Prohibit Password Reuse For A Minimum Of Five Generations

Invalid Finding:

Red Hat has updated the method for implementing this guidance. The `remember` token is now configured in the `/etc/security/pwhistory.conf` configuration file rather than the previously-used `/etc/pam.d/*` files.

If the scanner is flagging the `/etc/pam.d/*` files for lacking a `remember` token, this is a sign that the scanner's profiles need to be updated.

Proof of Correctness:

To validate that the required `remember=5` is present, execute:

```
$ grep -P '^(#|)remember' /etc/security/pwhistory.conf
```

The above *should* return a value similar to:

```
remember = 5
```

If the above has a null return, re-execute the `ash-linux.el8.STIGbyID.cat2.RHEL-08-pam_pwhistory` Saltstack state and re-validate.

The Installed Operating System Is Not Vendor Supported

Expected Finding:

This rule effects primarily "free" versions of the Red Hat Enterprise Linux operating system. This result is expected on the CentOS 8 – "Core" or "Stream" – Rocky and Alma linux distributions. Scanners that highlight this finding are looking for the presence of any *one* of the following RPMs:

- `redhat-release-client`
- `redhat-release-server`
- `redhat-release-workstation`
- `redhat-release-computenode`
- `redhat-release-virtualization-host`
- `oraclelinux-release`
- `sled-release`
- `sles-release`

And an `/etc/redhat-release` file with contents that aligns to one that's delivered with any of the preceding RPM. The various "free" versions of the Red Hat Enterprise Linux operating system will not have any of the above RPMs present.

If using a vendor-supported Linux and this scan finding occurs, it's likely that either the `release-` RPM is missing or damaged, something has unexpectedly altered the target's `/etc/redhat-release` file or the scanner is looking for a wildcarded `release` file under the `/etc` directory and there's an unexpected filename found.

All Remote Access Methods Must Be Monitored

Invalid Finding:

After execution of hardening-routines, the scanned-for `/etc/rsyslog.conf` entries *are* present. Any indications to the contrary are in error.

Proof of Correctness:

To verify that the necessary entries are present, execute:

```
grep -P '^(?!#)\s*([^\s]+)\s+/var/log/secure' /etc/rsyslog.conf
```

This will return output similar to the following:

```
authpriv.*  /var/log/secure
daemon.*    /var/log/secure
auth.*      /var/log/secure
```

Note: The above modification is made through the watchmaker-bundled Compliance as Code content. If the above entries are missing, ensure that the installed version of watchmaker is up to date, then rerun the `el8.VendorSTIG.remediate` state.

- The version of watchmaker may be checked by doing:

  ```
  /usr/local/bin/watchmaker --version
  ```

- The version of Compliance as Code installed may be checked by doing

```
# find /srv/watchmaker/salt/formulas/scap-formula/scap/content/guides/openscap -
↪type f | \
xargs grep '\s\s<xccdf.*version\supdate=' | \
sed
  -e 's/^.*latest">//'
  -e 's/<.*$//' | \
sort -u
```

All Content In A User's Home Directory Must Be Group-Owned By The Primary User

Expected Finding:

At initial scan, this finding is typically triggered by the installation of some standard "enterprise" services. Some of these services, due to how they execute, will create *some* of their files with `root` as the user- and/or (more importantly for this finding) group-owner.

The `oscap` content for this finding includes the caveat:

> *Due to OVAL limitation, this rule can report a false negative in a specific situation where two interactive users swap the group-ownership of folders or files in their respective home directories.*

While not a 100% overlap to the reason offered here, the caveat covers a common scenario. Other common scenarios may include:

- Unpacking of archive files authored on a different system
- Restoration of a user's `${HOME}` from another system to the current (scanned) system

In either of these further cases, such will most typically only show up on lifecycle scans and not provisioning-time scans

"Only Authorized Local User Accounts Exist on Operating System" is always flagged

Expected Finding:

Per the STIG notes:

> *Automatic remediation of this control is not available due to the unique requirements of each system.*

While-automation *could* be authored that would leverage a site- or host-specific allowed-users list to disable or delete forbidden accounts, there exists an extremely-high likelihood that scanners used against such configuration-controlled operating environments would not contain the scanning logic necessary to validate compliance. As such – and with or without user-controlling automation-content – STIG scanners would still flag systems that are *technically* compliant.

All Interactive User Home Directory Files Must Be Mode 0750 Or Less Permissive

Expected Finding:

Some scanners will erroneously alert on this for either/both of two reasons:

- The scanner is looking for files that have mode-zero for their "all" field regardless of owning-directory's mode-setting: in this case, the result is *technically* a correct finding but, from an *effective* security perspective is non-problematic
- The scanner may be confused if the "failed" file's group-permission is zero: in this case, the result is simply not valid

Add `nosuid` Option to `/boot`

Invalid Finding:

Some scanners will check to see what the mount-option is for the filesystem containing the `/boot` directory without first ensuring that `/boot` directory is actually a standalone filesystem. When `/boot` is not a standalone filesystem, it gets the same boot-options as the `/` filesystem and, therefore, cannot have the `nosuid` mount-option set.

Configure Multiple DNS Servers in `/etc/resolv.conf`

Expected Finding:

When deploying EL8 systems into environments with highly-available DNS servers, the system will typically only have *one* DNS server configured.

Enable Certmap in SSSD

Invalid Finding:

This finding is intended to result in a manual configuration-validation of the target system. Scanners that flag this finding typically include a note like:

> *Automatic remediation of this control is not available since all of the settings in the certmap need to be customized*

Further, configuration of the `sssd` certmap is typically required only for systems that are configured for *direct* authentication via client-certificate. This configuration-method is typically done only for systems with locally-attached SmartCard/PIV readers. "Remote" systems (such as those hosted with a CSP like AWS or Azure) typically *indirectly* authenticate with client-certificates (either through SSH key-forwarding or GSSAPI token-forwarding).

Verify that Shared Library Directories Have Root Ownership

Expected Finding:

Some applications and/or enterprise-integration tools may install private shared-libraries that are user- or group- owned by the installed-application. The scanner may identify these as insecure/improperly-owned, regardless of permission-setting on higherl-level directories.

Oracle Linux 8 STIGs Specify Conflicting `ClientAliveCountMax` values

Conflicting Guidance:

As of the time of this section's writing, there is a disagreement between the DISA STIG's target-value for the SSH daemon's `ClientAliveCountMax` value and that specified via the STIG's upstream content-project, Compliance As Code. The former specifies that the parameter's value should be `1`; the latter specifies that it should be `0`. This project's hardening implements the former as that is also the value specified by both the DISA STIG's and Compliance As Code project's recommended setting for Red Hat Linux 8.

Record Events When Privileged Executables Are Run

Invalid Finding:

Some security-scanners misidentify the compliance-state of target-systems for vulnerability-ID, V-248722 (OL08-00-030000). The relevant STIG check-text should be either or both of:

- `grep execve /etc/audit/audit.rules`
- `grep -r execve /etc/audit/rules.d`

After watchmaker is applied, the former returns:

```
-a always,exit -F arch=b32 -S execve -C gid!=egid -F key=setgid
-a always,exit -F arch=b64 -S execve -C gid!=egid -F key=setgid
-a always,exit -F arch=b32 -S execve -C uid!=euid -F key=setuid
-a always,exit -F arch=b64 -S execve -C uid!=euid -F key=setuid
```

While the latter returns:

```
/etc/audit/rules.d/setuid.rules:-a always,exit -F arch=b32 -S execve -C uid!=euid -F␣
↪key=setuid
/etc/audit/rules.d/setuid.rules:-a always,exit -F arch=b64 -S execve -C uid!=euid -F␣
↪key=setuid
/etc/audit/rules.d/setgid.rules:-a always,exit -F arch=b32 -S execve -C gid!=egid -F␣
↪key=setgid
/etc/audit/rules.d/setgid.rules:-a always,exit -F arch=b64 -S execve -C gid!=egid -F␣
↪key=setgid
```

It is the presence of the content in the file in the `/etc/audit/rules.d/` directory that results in – by way of the `augenrules` service – the presence of the correct content in the `/etc/audit/audit.rules` file.

EL 8 systems less than v8.4 must configure the password complexity module in the system-auth allow three retries or less

Invalid Finding:

This finding applies *only* to Enterprise Linux distros 8.0, 8.1, 8.2 and 8.3. As of the writing of this document all, properly-patched Enterprise Linux deployments are running 8.4 or higher. This finding does not apply to such systems

EL 8 must enable the hardware random number generator entropy gatherer service

Invalid Finding:

While this finding states that the `rngd` systemd unit must be enabled *and* active. Per the output from the `rngd.service` systemd unit:

```
$ systemctl status rngd
* rngd.service - Hardware RNG Entropy Gatherer Daemon
   Loaded: loaded (/usr/lib/systemd/system/rngd.service; enabled; vendor preset: enabled)
   Active: inactive (dead) since Tue 2023-06-27 15:21:25 UTC; 49s ago
Condition: start condition failed at Tue 2023-06-27 15:21:32 UTC; 42s ago
             ConditionKernelCommandLine=!fips=1 was not met
 Main PID: 214 (code=exited, status=0/SUCCESS)
```

The above-captured output's `ConditionKernelCommandLine`'s indication that the condition of `!fips=1` "was not met" means that this capability is not (currently) compatible with a system running with FIPS mode enabled. Enablement of FIPS mode is specified in another, earlier, higher-priority STIG-finding. As such, this setting will not be enableable while the higher-priority configuration-state is in place.

2.10 API Reference

2.10.1 watchmaker

Watchmaker module.

class watchmaker.**Arguments**(*config_path=None*, *log_dir=None*, *no_reboot=False*, *log_level=None*, **args*, ***kwargs*)

Bases: `dict`

Create an arguments object for the *watchmaker.Client*.

Parameters

- **config_path** – (`str`) Path or URL to the Watchmaker configuration file. If `None`, the default config.yaml file is used. (*Default*: `None`)
- **log_dir** – (`str`) Path to a directory. If set, Watchmaker logs to a file named `watchmaker.log` in the specified directory. Both the directory and the file will be created if necessary. If the file already exists, Watchmaker appends to it rather than overwriting it. If this argument evaluates to `False`, then logging to a file is disabled. Watchmaker will always output to stdout/stderr. Additionaly, Watchmaker workers may use this directory to keep other log files. (*Default*: `None`)
- **no_reboot** – (`bool`) Switch to control whether to reboot the system upon a successful execution of *watchmaker.Client.install()*. When this parameter is set, Watchmaker will suppress the reboot. Watchmaker automatically suppresses the reboot if it encounters an error. (*Default*: `False`)
- **log_level** – (`str`) Level to log at. Case-insensitive. Valid options include, from least to most verbose:
  - `critical`
  - `error`
  - `warning`
  - `info`
  - `debug`

Important: For all **Keyword Arguments**, below, the default value of `None` means Watchmaker will get the value from the configuration file. Be aware that `None` and `'None'` are two different values, with different meanings and effects.

Keyword Arguments

- **admin_groups** – (str) Set a salt grain that specifies the domain _groups_ that should have root privileges on Linux or admin privileges on Windows. Value must be a colon-separated string. On Linux, use the ^ to denote spaces in the group name. (*Default*: None)

```
admin_groups = "group1:group2"

# (Linux only) The group names must be lowercased. Also, if
# there are spaces in a group name, replace the spaces with a
# '^'.
admin_groups = "space^out"

# (Windows only) No special capitalization nor syntax
# requirements.
admin_groups = "Space Out"
```

- **admin_users** – (str) Set a salt grain that specifies the domain _users_ that should have root privileges on Linux or admin privileges on Windows. Value must be a colon-separated string. (*Default*: None)

```
admin_users = "user1:user2"
```

- **computer_name** – (str) Set a salt grain that specifies the computername to apply to the system. (*Default*: None)
- **environment** – (str) Set a salt grain that specifies the environment in which the system is being built. For example: dev, test, or prod. (*Default*: None)
- **salt_states** – (str) Comma-separated string of salt states to apply. A value of None will not apply any salt states. A value of 'Highstate' will apply the salt highstate. (*Default*: None)
- **ou_path** – (str) Set a salt grain that specifies the full DN of the OU where the computer account will be created when joining a domain. (*Default*: None)

```
ou_path="OU=Super Cool App,DC=example,DC=com"
```

- **extra_arguments** – (list) A list of extra arguments to be merged into the worker configurations. The list must be formed as pairs of named arguments and values. Any leading hypens in the argument name are stripped. (*Default*: [])

```
extra_arguments=['--arg1', 'value1', '--arg2', 'value2']

# This list would be converted to the following dict and merged
# into the parameters passed to the worker configurations:
{'arg1': 'value1', 'arg2': 'value2'}
```

class watchmaker.**Client**(*arguments*)

Bases: object

Prepare a system for setup and installation.

Keyword Arguments

arguments – (*Arguments*) A dictionary of arguments. See *watchmaker.Arguments*.

`install()`

Execute the watchmaker workers against the system.

Upon successful execution, the system will be properly provisioned, according to the defined configuration and workers.

watchmaker.managers

Watchmaker managers module.

watchmaker.managers.platform

Watchmaker base manager.

`class watchmaker.managers.platform.PlatformManagerBase(system_params, *args, **kwargs)`

Bases: `object`

Base class for operating system managers.

All child classes will have access to methods unless overridden by an identically-named method in the child class.

Parameters

system_params – (`dict`) Attributes, mostly file-paths, specific to the system-type (Linux or Windows). The dict keys are as follows:

prepdir:
Directory where Watchmaker will keep files on the system.

readyfile:
Path to a file that will be created upon successful completion.

logdir:
Directory to store log files.

workingdir:
Directory to store temporary files. Deleted upon successful completion.

restart:
Command to use to restart the system upon successful completion.

shutdown_path:
(Windows-only) Path to the Windows `shutdown.exe` command.

`retrieve_file(url, filename)`

Retrieve a file from a provided URL.

Supports all `urllib.request` handlers, as well as S3 buckets.

Parameters

- **url** – (`str`) URL to a file.
- **filename** – (`str`) Path where the file will be saved.

`create_working_dir(basedir, prefix)`

Create a directory in `basedir` with a prefix of `prefix`.

Parameters

- **prefix** – (`str`) Prefix to prepend to the working directory.

- **basedir** – (str) The directory in which to create the working directory.

Returns
Path to the working directory.

Return type
str

`call_process`(*cmd*, *log_pipe='all'*, *raise_error=True*)

Execute a shell command.

Parameters

- **cmd** – (list) Command to execute.
- **log_pipe** – (str) Controls what to log from the command output. Supports three values: `stdout`, `stderr`, `all`. (*Default*: `all`)
- **raise_error** – (bool) Switch to control whether to raise if the command return code is non-zero. (*Default*: `True`)

Returns
Dictionary containing three keys: `retcode` (int), `stdout` (bytes), and `stderr` (bytes).

Return type
dict

`cleanup`()

Delete working directory.

`extract_contents`(*filepath*, *to_directory*, *create_dir=False*)

Extract a compressed archive to the specified directory.

Parameters

- **filepath** – (str) Path to the compressed file. Supported file extensions:
  - *.zip*
  - *.tar.gz*
  - *.tgz*
  - *.tar.bz2*
  - *.tbz*
- **to_directory** – (str) Path to the target directory
- **create_dir** – (bool) Switch to control the creation of a subdirectory within `to_directory` named for the filename of the compressed file. (*Default*: `False`)

class watchmaker.managers.platform.**LinuxPlatformManager**(*system_params*, **args*, ***kwargs*)

Bases: *PlatformManagerBase*

Base class for Linux Platforms.

Serves as a foundational class to keep OS consistency.

class watchmaker.managers.platform.**WindowsPlatformManager**(*system_params*, **args*, ***kwargs*)

Bases: *PlatformManagerBase*

Base class for Windows Platform.

Serves as a foundational class to keep OS consistency.

watchmaker.managers.worker_manager

Watchmaker workers manager.

class watchmaker.managers.worker_manager.**WorkersManagerBase**(*system_params*, *workers*, **args*, ***kwargs*)

Bases: object

Base class for worker managers.

Parameters

- **system_params** – (dict) Attributes, mostly file-paths, specific to the system-type (Linux or Windows).
- **workers** – (collections.OrderedDict) Workers to run and associated configuration data.

worker_cadence()

Manage worker cadence.

class watchmaker.managers.worker_manager.**LinuxWorkersManager**(*system_params*, *workers*, **args*, ***kwargs*)

Bases: *WorkersManagerBase*

Manage the worker cadence for Linux systems.

cleanup()

Execute cleanup function.

class watchmaker.managers.worker_manager.**WindowsWorkersManager**(*system_params*, *workers*, **args*, ***kwargs*)

Bases: *WorkersManagerBase*

Manage the worker cadence for Windows systems.

cleanup()

Execute cleanup function.

watchmaker.workers

Watchmaker workers module.

watchmaker.workers.base

Watchmaker base worker.

class watchmaker.workers.base.**WorkerBase**(*system_params*, **args*, ***kwargs*)

Bases: object

Define the architecture of a Worker.

abstract before_install()

Add before_install method to all child classes.

abstract install()

Add install method to all child classes.

watchmaker.workers.salt

Watchmaker salt worker.

class watchmaker.workers.salt.**SaltBase**(**args*, ***kwargs*)

Bases: *WorkerBase*, *PlatformManagerBase*

Cross-platform worker for running salt.

Parameters

- **salt_debug_log** – (list) Filesystem path to a file where the salt debug output should be saved. When unset, the salt debug log is saved to the Watchmaker log directory. (*Default*: '')
- **salt_content** – (str) URL to a salt content archive (zip file) that will be uncompressed in the watchmaker salt "srv" directory. This typically is used to create a top.sls file and to populate salt's file_roots. (*Default*: '')
  - *Linux*: /srv/watchmaker/salt
  - *Windows*: C:\Watchmaker\Salt\srv
- **salt_content_path** – (str) Used in conjunction with the "salt_content" arg. Glob pattern for the location of salt content files inside the provided salt_content archive. To be used when salt content files are located within a sub-path of the archive, rather than at its top-level. Multiple paths matching the given pattern will result in error. E.g. salt_content_path='*/' (*Default*: '')
- **salt_states** – (str) Comma-separated string of salt states to execute. When "highstate" is included with additional states, "highstate" runs first, then the other states. Accepts two special keywords (case-insensitive): (*Default*: 'highstate')
  - none: Do not apply any salt states.
  - highstate: Apply the salt "highstate".
- **exclude_states** – (str) Comma-separated string of states to exclude from execution. (*Default*: '')
- **user_formulas** – (dict) Map of formula names and URLs to zip archives of salt formulas. These formulas will be downloaded, extracted, and added to the salt file roots. The zip archive must contain a top-level directory that, itself, contains the actual salt formula. To "overwrite" bundled submodule formulas, make sure the formula name matches the submodule name. (*Default*: {})
- **admin_groups** – (str) Sets a salt grain that specifies the domain groups that should have root privileges on Linux or admin privileges on Windows. Value must be a colon-separated string. E.g. "group1:group2" (*Default*: '')
- **admin_users** – (str) Sets a salt grain that specifies the domain users that should have root privileges on Linux or admin privileges on Windows. Value must be a colon-separated string. E.g. "user1:user2" (*Default*: '')
- **environment** – (str) Sets a salt grain that specifies the environment in which the system is being built. E.g. dev, test, prod, etc. (*Default*: '')
- **ou_path** – (str) Sets a salt grain that specifies the full DN of the OU where the computer account will be created when joining a domain. E.g. "OU=SuperCoolApp,DC=example,DC=com" (*Default*: '')

- **pip_install** – (list) Python packages to be installed prior to applying the high state. (*Default*: `[]`)
- **pip_args** – (list) Options to pass to pip when installing packages. (*Default*: `[]`)
- **pip_index** – (str) URL used for an index by pip. (*Default*: `https://pypi.org/simple`)

`before_install()`

Validate configuration before starting install.

`install()`

Install Salt.

`run_salt(`*command*, ***kwargs*`)`

Execute salt command.

Parameters
command – (str or list) Salt options and a salt module to be executed by salt-call. Watchmaker will always begin the command with the options `--local`, `--retcode-passthrough`, and `--no-color`, so do not specify those options in the command.

`service_status(`*service*`)`

Get the service status using salt.

Parameters
service – (obj:*str*) Name of the service to query.

Returns

`('running', 'enabled')`
First element is the service running status. Second element is the service enabled status. Each element is a `bool` representing whether the service is running or enabled.

Return type
tuple

`service_stop(`*service*`)`

Stop a service status using salt.

Parameters
service – (str) Name of the service to stop.

Returns
`True` if the service was stopped. `False` if the service could not be stopped.

Return type
bool

`service_start(`*service*`)`

Start a service status using salt.

Parameters
service – (str) Name of the service to start.

Returns
`True` if the service was started. `False` if the service could not be started.

Return type
bool

`service_disable`(*service*)

Disable a service using salt.

Parameters
service – (`str`) Name of the service to disable.

Returns
`True` if the service was disabled. `False` if the service could not be disabled.

Return type
`bool`

`service_enable`(*service*)

Enable a service using salt.

Parameters
service – (`str`) Name of the service to enable.

Returns
`True` if the service was enabled. `False` if the service could not be enabled.

Return type
`bool`

`process_grains`()

Set salt grains.

`process_states`(*states*, *exclude*)

Apply salt states but exclude certain states.

Parameters

- **states** – (`str`) Comma-separated string of salt states to execute. When "highstate" is included with additional states, "highstate" runs first, then the other states. Accepts two special keywords (case-insensitive):
  - `none`: Do not apply any salt states.
  - `highstate`: Apply the salt "highstate".
- **exclude** – (`str`) Comma-separated string of states to exclude from execution.

`class watchmaker.workers.salt.SaltLinux`(**args*, ***kwargs*)

Bases: *SaltBase*, *LinuxPlatformManager*

Run salt on Linux.

Parameters

- **install_method** – (`str`) **Required**. Method to use to install salt. (*Default*: `yum`)
  - `yum`: Install salt from an RPM using yum.
  - `git`: Install salt from source, using the salt bootstrap.
- **bootstrap_source** – (`str`) URL to the salt bootstrap script. Required if `install_method` is `git`. (*Default*: `''`)
- **git_repo** – (`str`) URL to the salt git repo. Required if `install_method` is `git`. (*Default*: `''`)
- **salt_version** – (`str`) A git reference present in `git_repo`, such as a commit or a tag. If not specified, the HEAD of the default branch is used. (*Default*: `''`)

install()

Install salt and execute salt states.

class watchmaker.workers.salt.**SaltWindows**(**args*, ***kwargs*)

Bases: *SaltBase*, *WindowsPlatformManager*

Run salt on Windows.

Parameters

- **installer_url** – (str) **Required**. URL to the salt installer for Windows. (*Default*: '')
- **ash_role** – (str) Sets a salt grain that specifies the role used by the ash-windows salt formula. E.g. "MemberServer", "DomainController", or "Workstation" (*Default*: '')

install()

Install salt and execute salt states.

watchmaker.workers.yum

Watchmaker yum worker.

class watchmaker.workers.yum.**Yum**(**args*, ***kwargs*)

Bases: *WorkerBase*, *LinuxPlatformManager*

Install yum repos.

Parameters

repo_map – (list) List of dictionaries containing a map of yum repo files to systems. (*Default*: [])

get_dist_info()

Validate the Linux distro and return info about the distribution.

get_mapped_dist_name()

Return a normalized dist-name value.

before_install()

Validate configuration before starting install.

install()

Install yum repos defined in config file.

2.11 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

2.11.1 Bug Reports

When reporting a bug please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

2.11.2 Documentation Improvements

Watchmaker could always use more documentation, whether as part of the official Watchmaker docs, in docstrings, or even on the web in blog posts, articles, and such. The official documentation is maintained within this project in docstrings or in the docs directory. Contributions are welcome, and are made the same way as any other code. See *Development* guide.

2.11.3 Feature Requests and Feedback

The best way to send feedback is to file an issue on GitHub.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a community-driven, open-source project, and that code contributions are welcome. :)

2.11.4 Development Guide

To set up `watchmaker` for local development:

1. Fork watchmaker (look for the "Fork" button).
2. Clone your fork locally and update the submodules:

   ```
   git clone https://github.com/your_name_here/watchmaker.git && cd watchmaker
   git submodule update --init --recursive
   ```

3. Create a branch for local development:

   ```
   git checkout -b name-of-your-bugfix-or-feature
   ```

4. Now you can make your changes locally.
5. When you're done making changes, use tox to run the linter, the tests, and the doc builder:

   ```
   tox
   ```

   NOTE: This will test the package in all versions of Python listed in `tox.ini`. If `tox` cannot find the interpreter for the version, the test will fail with an `InterpreterNotFound` error. This is ok, as long as at least one interpreter runs and the tests pass. You can also specify which *tox environments* to execute, which can be used to restrict the Python version required.

   You can also rely on Travis and Appveyor to run the tests after opening the pull request. They will be slower though...

6. In addition to building the package and running the tests, `tox` will build any docs associated with the change. They will be located in the `dist/docs` directory. Navigate to the folder, open index.html in your browser, and verify that the doc text and formatting are as you intended.

   If you *only* want to build the docs, run:

   ```
   tox -e docs
   ```

   Note: depending on your development environment, your browser may not be able to locate to above-created, rendered content. If so, it will be necessary to copy or relocate the documentation to location that your preferred-browser *can* access them.

7. Commit your changes and push your branch to GitHub:

   ```
   git add .
   git commit -m "Your detailed description of your changes."
   git push origin name-of-your-bugfix-or-feature
   ```

8. Submit a pull request through the GitHub website.

2.11.5 Pull Request Guidelines

If you need some code review or feedback while you are developing the code just open the pull request.

For pull request acceptance, you should:

1. Include passing tests (Ensure `tox` is successful).
2. Update documentation whenever making changes to the API or functionality.
3. Add a note to `CHANGELOG.md` about the changes. Include a link to the pull request.

2.11.6 Tox Tips

1. The *primary* tox environments for `watchmaker` include:
   - `check`
   - `docs`
   - `py26`
   - `py27`
   - `py35`
   - `py36`
2. To run a subset of environments:

   ```
   tox -e <env1>,<env2>,<env3>,etc
   ```

3. To run a subset of tests:

   ```
   tox -e <environment> -- py.test -k <test_myfeature>
   ```

4. To run all the test environments in *parallel*, use `detox`:

```
pip install detox
detox
```

2.11.7 Build a Development Branch in EC2

To install and run a development branch of watchmaker on a new EC2 instance, specify something like this for EC2 userdata:

- **For Linux**: Modify `GIT_REPO` and `GIT_BRANCH` to reflect working values for your development build. Modify `PIP_URL` and `PYPI_URL` as needed.

```
#!/bin/sh
GIT_REPO=https://github.com/<your-github-username>/watchmaker.git
GIT_BRANCH=<your-branch>

PYPI_URL=https://pypi.org/simple

# Setup terminal support for UTF-8
export LC_ALL=en_US.UTF-8
export LANG=en_US.UTF-8

# Install pip
python3 -m ensurepip

# Install git
yum -y install git

# Upgrade pip and setuptools
python3 -m pip install --index-url="$PYPI_URL" --upgrade pip setuptools

# Clone watchmaker
git clone "$GIT_REPO" --branch "$GIT_BRANCH" --recursive

# Install watchmaker
cd watchmaker
python3 -m pip install --index-url "$PYPI_URL" --editable .

# Run watchmaker
watchmaker --log-level debug --log-dir=/var/log/watchmaker
```

- **For Windows**: Modify `GitRepo` and `GitBranch` to reflect working values for your development build. Optionally modify `BootstrapUrl`, `PythonUrl`, `GitUrl`, and `PypiUrl` as needed.

```
<powershell>
$GitRepo = "https://github.com/<your-github-username>/watchmaker.git"
$GitBranch = "<your-branch>"

$BootstrapUrl = "https://watchmaker.cloudarmor.io/releases/latest/watchmaker-
↪bootstrap.ps1"
$PythonUrl = "https://www.python.org/ftp/python/3.10.11/python-3.10.11-amd64.exe"
$GitUrl = "https://github.com/git-for-windows/git/releases/download/v2.40.1.windows.
↪1/Git-2.40.1-64-bit.exe"
```

(continues on next page)

(continued from previous page)

```
$PypiUrl = "https://pypi.org/simple"

# Use TLS 1.2+
[Net.ServicePointManager]::SecurityProtocol = "Tls12, Tls13"

# Download bootstrap file
$BootstrapFile = "${Env:Temp}\$(${BootstrapUrl}.split("/")[-1])"
(New-Object System.Net.WebClient).DownloadFile($BootstrapUrl, $BootstrapFile)

# Install python and git
& "$BootstrapFile" `
    -PythonUrl "$PythonUrl" `
    -GitUrl "$GitUrl" `
    -Verbose -ErrorAction Stop

# Upgrade pip and setuptools
python -m pip install --index-url="$PypiUrl" --upgrade pip setuptools

# Clone watchmaker
git clone "$GitRepo" --branch "$GitBranch" --recursive

# Install watchmaker
cd watchmaker
python -m pip install --index-url "$PypiUrl" --editable .

# Run watchmaker
watchmaker --log-level debug --log-dir=C:\Watchmaker\Logs
</powershell>
```

2.12 Changelog

2.12.1 0.28.2

Released: 2023.10.31

Summary:

- Updates Watchmaker default config to use Salt 3006.4
- Documents invalid finding in EL8 for remote access monitoring methods
- ash-linux-formula
  - Addresses several EL8 Cat2 findings from recent SCAP scans
- join-domain-formula
  - (Linux) Adds cron config that refreshes AD computer object attributes

2.12.2 0.28.1

Released: 2023.10.05

Summary:

- Fixes clobbering of `computer-name` grain when `computer-name-pattern` is also provided. This prevented the `name-computer-formula` from setting the name specified by the user
- Updates FAQ to include vendor guidance for EL8.8+
- Adds guidance on OpenSSH key signing requirements for EL8
- ash-linux-formula
  - Adds handler to address pam faillock findings on EL8

2.12.3 0.28.0

Released: 2023.09.14

Summary:

- Add watchmaker config argument `computer_name_pattern`, and exit with error if provided `computer_name` does not match. Also writes grain for use with name-computer-formula
- Updates default watchmaker config to use salt 3006.2
- Documents customization options for the watchmaker salt content
- Documents workarounds for known "gotchas" when applying EL7 and EL8 STIG controls
- ash-linux-formula
  - Supports customization for mapping users to different SELinux contexts
  - Removes el7 and EL8 STIG handlers that are now provided by SCAP remediation content
  - Consolidates all separate EL8 PAM handlers to states based on new authselect capabilities
- join-domain-formula
  - Adds support for `tries` option that retries a failed join domain action
  - Integrates with ash-linux PAM handlers to apply STIG controls, if available
- trellix-agent-formula
  - Refactors firewalld states around newer salt functionality
- name-computer-formula
  - Supports reading pattern from salt grain

2.12.4 0.27.5

Released: 2023.08.07

Summary:

- Adds doc section on troubleshooting Watchmaker, to include common errors, issues, and relevant log files
- Updates AWS provider to support EC2 instances configured for only IMDSv2
- ash-linux-formula
  - Addresses additional STIG findings for EL7 and EL8
- join-domain-formula
  - Resolves issue with collision detection when deploying a new system with a hostname that already exists in the domain
  - Corrects usage of StartTLS when searching for a computer object in the domain
  - Provides several new options for controlling whether TLS is used when searching for a computer object in the domain, and whether an error will be treated as fatal or not

2.12.5 0.27.4

Released: 2023.06.28

Summary:

- Updates guidance on Linux STIG findings relating to SELinux context and sudo privilege escalation
- ash-linux-formula
  - Adds additional guidance on pillar content usage
  - Adds additional EL7 STIG handlers
  - Removes duplicate EL7 STIG handlers for audit rules
- forescout-secure-connector-formula
  - Adds state to ensure correct directory ownership
- join-domain-formula
  - Updates sssd to support a variety of conf parameters
- scap-formula
  - Updates DISA SCAP content

2.12.6 0.27.3

Released: 2023.05.25

Summary:

- Fixes issue with standalone binary on FIPS-enabled EL8 systems, by packaging libcrypto and libssl libraries in the binary

2.12.7 0.27.2

Released: 2023.05.18

Summary:

- Adds support for salt 3006
- Builds standalone executable using Python 3.10
- Documents additional expected findings for EL8 systems
- Uses Python 3.10 in all documentation references
- Updates default config to use salt 3006.1
- Uses SCC 5.7.1 in default salt content
- ash-linux-formula
  - Simplifies logic for managing faillock.conf
- ash-windows-formula
  - Updates custom modules for compatibility with Salt 3006 while remaining backwards compatible with salt 3005 and earlier
- splunkforwarder-formula
  - Sets splunk user/group on files and directories, eliminating "Changes" when re-executing the formula

2.12.8 0.27.1

Released: 2023.05.08

Summary:

- Fixes typo in upload of Windows standalone binary to GitHub Releases
- Documents known/spurious EL8 findings that scanning utilities may flag erroneously
- Fixes the check that skips reinstalling salt when the correct version is already installed
- Publishes EL8 scap scans as a release artifact to `watchmaker.cloudarmor.io`, alongside the standalone binaries
- Updates scap pillar in default salt content to run scans properly on CentOS Stream and scap version 1.3
- ash-linux-formula
  - Fixes oscap remediation on CentOS Stream 8 and Oracle Linux 8
  - Addresses numerous additional STIG findings on EL8 systems that were not addressed with oscap remediation
  - Attempts to address EL8 issue with aws-cli, where fapolicyd blocks execution
- forescout-secure-connector-formula
  - Establishes symlink so logs are written to `/var/log` partition
- scap-formula
  - Updates openscap content to v0.1.67, using scap 1.3 datastreams. This also addresses issues with expiry on passwordless local users

2.12.9 0.27.0

Released: 2023.03.31

Summary:

- Releases support for EL8 platforms, to include Red Hat 8, CentOS 8 Stream, and Oracle Linux 8. Future work may also add support for Rocky Linux 8 and Alma Linux 8
  - CAVEAT: With this release, on FIPS-enabled EL8 systems, please use the PyPi install or the source install methods. Currently, the standalone method for EL8 **does not** work when the system is FIPS-enabled. The problem is not yet entirely understood. Further investigation is needed before this issue can be resolved
  - UPDATE: The issue with FIPS-enabled EL8 and the standalone binary is fixed in Watchmaker 0.27.3
- Updates salt worker to avoid re-installing salt when `salt-call --version` matches the `salt_version` in the Watchmaker config
- Updates EL7 findings documentation to line up with latest stig version
- Installs `dnspython` package when using default Watchmaker config, to support the join-domain `nsupdate` state
- ash-linux-formula
  - Adds handlers to address findings in latest stig versions and increase coverage
- mcafee-agent-formula
  - Adds a `trellix-agent` salt state to support the new name for the software
- join-domain-formula
  - Linux: Adds an `nsupdate` salt state that will register forward and reverse dns records
  - Windows: Updates collision handling and join actions to use the same domain controller
  - Windows: Supports collision handling where an existing computer object was created by a different service account than is now specified for the join action
- winrepo: Adds a `trellix-agent` package definition

2.12.10 0.26.5

Released: 2023.03.10

Summary:

- join-domain-formula
  - Linux: Output journald logs on join-domain failures
  - Linux: Re-order sssd conf file Salt states and explicitly set replace setting to false
  - Linux: Patch find-collision.sh script to fix computer-object search

2.12.11 0.26.4

Released: 2023.03.03

Summary:

- Attempts to fix the release automation so the Windows standalone is published to GitHub Releases
- Validates functionality with salt 3005.1 and updates default config to use salt 3005.1
- join-domain-formula
  - Windows: Provides pillar options to configure DNS registration settings, to support registration of reverse DNS records

2.12.12 0.26.3

Released: 2023.02.27

Summary:

- Skips provider detection when provider requirements are not installed
- Updates watchmaker salt log config to avoid capturing senstive data in salt log
- forescout-secure-connector-formula
  - Adds support for EL8 when FIPS is enabled
- name-computer-formula
  - Sets hostname as fqdn when `dns_domain` is provided
- join-domain-formula
  - Runs fix-collision script when using sssd
  - Updates fix-collision to avoid capturing sensitive values in salt log
  - Updates sssd method to set extra os attributes only when requested
  - Updates windows join script to avoid capturing sensitive values in salt log

2.12.13 0.26.2

Released: 2023.02.13

Summary:

- Fixes publishing of Windows standalone to GitHub Releases
- docs
  - Provides guidance on using S3 URL feature in config references
  - Describes prerequisites for using AWS and Azure features
  - Removes references to EL6 and Python 2.6
  - Removes references to deprecated `--s3-url` argument
- join-domain-formula
  - Adds support for EL8, using `sssd` to perform the domain-join

2.12.14 0.26.1

Released: 2023.02.08

Summary:

- Uses pyinstaller directly to build standalone packages, eliminating dependency on gravitybee
- Uses new python apis to reference package metadata and resources, improving support for alternative packaging methods, like in-memory runtimes (pyoxidizer) or ziparchives
- Adds PEP517 package metadata
- [Alpha] Allows watchmaker to run on Red Hat Enterprise Linux 8, Centos 8 Stream, Oracle Linux 8, Alma Linux 8, and Rocky Linux 8. Currently on the ash-linux hardening formula will work; none of the other salt formulas have yet been updated for EL8 support
- ash-windows
  - Fixes warning in `lgpo` module about using `is` instead of == to compare string-literal values

2.12.15 0.26.0

Commit Delta: Change from 0.25.0 release

Released: 2022.12.21

Summary:

- Adds support for posting to a status provider. Initial capability supports AWS and will post the Watchmaker status to an EC2 instance tag. Status values include "Running", "Completed", or "Failed". For more information on this feature, see https://watchmaker.cloudarmor.io/en/stable/configuration.html#status.
- [Alpha] Support posting status to Azure as a Virtual Machine tag
- [Alpha] Support for EL8 platforms is improving but still in development. Targeted platforms include: Red Hat Enterprise Linux 8, Centos 8 Stream, Oracle Linux 8, Alma Linux 8, and Rocky Linux 8
- ash-linux-formula
  - Supports EL8 platforms
- join-domain-formula
  - Fixes hostname logic so automatic renaming works correctly
- scap-formula
  - Supports EL8 platforms

2.12.16 0.25.0

Commit Delta: Change from 0.24.3 release

Released: 2022.10.05

Summary:

- [Alpha] Begins initial preparation to support running watchmaker on EL8 platforms
- forescout-secure-connector-formula
  - First release that packages a formula for ForeScout Secure Connector

2.12.17 0.24.3

Commit Delta: Change from 0.24.2 release

Released: 2022.09.16

Summary:

- ash-linux-formula
  - Adds check to ensure root account password is set to not expire
- join-domain-formula
  - Removes PAM Lsass login re-configuration

2.12.18 0.24.2

Commit Delta: Change from 0.24.1 release

Released: 2022.08.16

Summary:

- scap-formula
  - Updates OpenSCAP and DISA STIG content
  - Linux: Adds EL8 content
- watchmaker-salt-content
  - Windows: Re-adds scap scan using public distributable SCC 5.5

2.12.19 0.24.1

Commit Delta: Change from 0.24.0 release

Released: 2022.07.27

Summary:

- Builds Linux standalone with python 3.8
- Updates default config to use Salt 3004.2
- Updates Windows usage docs with requirement to enforce modern TLS versions
- join-domain-formula
  - Windows: Quotes path when running scripts to configure local admin group
- ash-linux-formula
  - Tests if grub.cfg exists before attempting to modify it

2.12.20 0.24.0

Commit Delta: Change from 0.23.4 release

Released: 2022.03.09

Summary:

- Updates default config to use Salt 3004
- scap-formula
  - Fixes invalid requisite reference for scap.scan

2.12.21 0.23.4

Commit Delta: Change from 0.23.3 release

Released: 2021.12.20

Summary:

- ash-windows: Adds baseline `ash-windows.cis_1_3_0`
- Builds python 3.8 into standalone binary instead of python 3.6
- Uses SERVER_AUTH for ssl context, fixing bug resulting from incorrect use of CLIENT_AUTH previously

2.12.22 0.23.3

Commit Delta: Change from 0.23.2 release

Released: 2021.09.28

Summary:

- Added publishing of SCAP reports for Linux systems with each release
- Fixed CLI behavior when passing 'none' value, e.g. `--salt-states none`
- Updated default config to use Salt 3003.3
- mcafee-agent-formula
  - (Linux) Refactored to allow install over an existing installation

2.12.23 0.23.2

Commit Delta: Change from 0.23.1 release

Released: 2021.08.11

Summary:

- Patched Salt worker to be case-sensitive when processsing Salt states
- Refactored Salt states handling between CLI and config.yaml
- Standalone packages are now based on EL7 and are no longer compatible with EL6 (EL6 hasn't been supported for a while now)

2.12.24 0.23.1

Commit Delta: Change from 0.23.0 release

Released: 2021.07.15

Summary:

- ash-linux-formula
  - Supports managing FIPS when / and /boot are on the same partition
  - Allows `oscap remediate` to exit non-zero on valid errors
- Supports parsing extra_arguments when passed using = as the separator
  - E.g. `--user-formulas='{"foo-formula": "https://url-to/foo-formula.zip"}'`

2.12.25 0.23.0

Commit Delta: Change from 0.22.2 release

Released: 2021.07.08

Summary:

- Adds capability to run extra states after highstate
  - E.g. from cli, `--salt-states highstate,foo,bar`
  - E.g. in config file, `salt_states: highstate,foo,bar`
- Adds capability to pass complex worker arguments on the cli as JSON or YAML
  - E.g. `--user-formulas '{"foo-formula": "https://url-to/foo-formula.zip"}'`

2.12.26 0.22.2

Commit Delta: Change from 0.22.1 release

Released: 2021.06.17

Summary:

- ash-linux-formula
  - Patches RHEL-07-040810 to apply to only iptables-services RPM and not core-package iptables RPM

2.12.27 0.22.1

Commit Delta: Change from 0.22.0 release

Released: 2021.05.11

Summary:

- ash-linux-formula/nessus-agent-formula
  - Patches maxdepth parameter to use integer type to support Jinja rendering in Salt 3003

2.12.28 0.22.0

Commit Delta: Change from 0.21.9 release

Released: 2021.05.07

Summary:

- Updates default config.yaml to use Salt 3003
- ash-linux-formula
  - Adds ability to selectively skip extra EL7 STIG handlers
- nessus-agent-formula
  - (Linux) Updates nessus-agent to call install and configure states

2.12.29 0.21.9

Commit Delta: Change from 0.21.8 release

Released: 2021.04.26

Summary:

- Provides support for Salt 3003
- ash-linux-formula
  - Updates syntax to support Salt 3003
  - RHEL-07-040160 - Ensure no (competing) attempts to set TMOUT
  - RHEL-07-040860 - Adds ability to handle lack of /etc/sysct.conf file
- nessus-agent-formula
  - Separate agent install and configuration to support baked-in Nessus agent installations
- join-domain-formula
  - (Windows) Add double-quotes to Members parameter in order for startup task state to work with Salt 3003

2.12.30 0.21.8

Commit Delta: Change from 0.21.7 release

Released: 2021.03.11

Summary:

- Updated CI configs to set the correct version for the Windows standalone package. Effectively, this version is the same as 0.21.7.

2.12.31 0.21.7

Commit Delta: Change from 0.21.6 release

Released: 2021.03.10

Summary:

- ash-linux-formula
  - Coordinates `sshd` service restarts across all states that modify `/etc/sshd_config`, so the service restarts only once. This avoids systemd failures when the service restarts too frequently. See ash-linux-formula PR #303.

2.12.32 0.21.6

Commit Delta: Change from 0.21.5 release

Released: 2021.03.03

Summary:

- ash-linux-formula
  - Adds patch to re-enable NOPASSWD sudo for users in /etc/sudoers.d/ after oscap remediation.

2.12.33 0.21.5

Commit Delta: Change from 0.21.4 release

Released: 2021.02.25

Summary:

- ash-linux-formula
  - Replace `watch` with `listen` to restart the `sshd` service a single time
  - Make state RHEL-07-040560 more resilient when the yum group info is missing
- scap-formula
  - Updates SCAP content from DISA (as of February 2021) and OpenSCAP (v0.1.54)
- Update watchmaker default `config.yaml` to use salt v2019.2.8
- Ability to browse Watchmaker Cloudarmor repo

2.12.34 0.21.4

Commit Delta: Change from 0.21.3 release

Released: 2020.12.04

Summary:

- nessus-agent-formula
  - (Linux) Switch to using Salt service state to ensure Nessus agent service is running

2.12.35 0.21.3

Commit Delta: Change from 0.21.2 release

Released: 2020.10.26

Summary:

- watchmaker-salt-content
  - (Linux) Updates scap-formula pillar to use alternative stig profile parameter for Red Hat

2.12.36 0.21.2

Commit Delta: Change from 0.21.1 release

Released: 2020.10.05

Summary:

- (Windows) Removes winrepo.genrepo usage in Salt worker since it's no longer required

2.12.37 0.21.1

Commit Delta: Change from 0.21.0 release

Released: 2020.08.20

Summary:

- splunkforwarder-formula
  - (Linux) Patches splunkforwarder state to work with Splunk Universal Forwarder v7.3.6

2.12.38 0.21.0

Commit Delta: Change from 0.20.5 release

Released: 2020.08.12

Summary:

- Updates default watchmaker config.yaml to use salt 2019.2.5

2.12.39 0.20.5

Commit Delta: Change from 0.20.4 release

Released: 2020.07.16

Summary:

- splunkforwarder-formula
  - (Linux) Patches splunkforwarder state to work with salt 2019.2.5

2.12.40 0.20.4

Commit Delta: Change from 0.20.3 release

Released: 2020.07.15

Summary:

- splunkforwarder-formula
  - (Windows) Patches splunkforwarder state to work with salt 2019.2.5

2.12.41 0.20.3

Commit Delta: Change from 0.20.2 release

Released: 2020.07.07

Summary:

- join-domain-formula
  - (Linux) Fixes issue with admin users not being able to sudo

2.12.42 0.20.2

Commit Delta: Change from 0.20.1 release

Released: 2020.07.01

Summary:

- scap-formula
  - Updates SCAP content from DISA (as of June 2020) and OpenSCAP (v0.1.50)

2.12.43 0.20.1

Commit Delta: Change from 0.20.0 release

Released: 2020.05.19

Summary:

- ash-linux-formula
  - Fixes issue with Postfix occasionally failing to start

2.12.44 0.20.0

Commit Delta: Change from 0.19.0 release

Released: 2020.05.06

Summary:

- Adds capability to install Python packages using Pip in Salt's Python interpreter

2.12.45 0.19.0

Commit Delta: Change from 0.18.2 release

Released: 2020.05.01

Summary:

- Updates Watchmaker file permissions and makes them more restrictive
- Adds new SaltWorker optional argument `--salt-content-path` that allows specifying glob pattern for salt files located within salt archive file

2.12.46 0.18.2

Commit Delta: Change from 0.18.1 release

Released: 2020.04.02

Summary:

- vault-auth-formula
  - Rename state to vault-auth
  - Add url keyword argument to read_secret execution module

2.12.47 0.18.1

Commit Delta: Change from 0.18.0 release

Released: 2020.03.23

Summary:

- Updates version constraint in default config to allow newer versions

2.12.48 0.18.0

Commit Delta: Change from 0.17.5 release

Released: 2020.03.23

Summary:

- Removes deprecated `emet-formula` and `dotnet4-formula` submodules
- Adds new `vault-auth-formula` submodule
- ash-windows-formula
  - Replaces usage of Apply_LGPO_Delta.exe with native python and salt functionality
  - Addresses additional findings for domain-joined systems
  - Removes deprecated baselines from Windows Server 2008 R2, 8.1, and IE 8, 9, and 10

2.12.49 0.17.5

Commit Delta: Change from 0.17.4 release

Released: 2020.03.13

Summary:

- join-domain-formula
  - Allow use of password with Linux join domain capability

2.12.50 0.17.4

Commit Delta: Change from 0.17.3 release

Released: 2020.02.28

Summary:

- ash-linux-formula
  - Updates custom STIGbyID baseline to address several scan findings.
- Add content for RHEL-07-040530/SV-86899

2.12.51 0.17.3

Commit Delta: Change from 0.17.2 release

Released: 2020.02.26

Summary:

- dotnet4-formula
  - Fix compatibility with Windows Server 2019 by using 2019 hotfixes
- ash-linux-formula
  - Improvements for STIGv2r6
  - Fix collisions caused by cat2 IDs and DISA numbering change
  - Use Salt Stack version 2019.2-2

2.12.52 0.17.2

Commit Delta: Change from 0.17.1 release

Released: 2020.02.25

Summary:

- Documents configuration vs cli argument handling and precedence
- Provides a table mapping common scan findings to an associated Finding ID
- Restores propagation of the `None` value on the cli to the workers
- ash-linux-formula
  - Ensures aide configuration complies with FIPS requirements

- ash-windows-formula
  - Adds missing sls to restore support for Windows 10
- join-domain-formula
  - Suppresses join-domain command in salt log output
  - (Windows) Supports using salt-native pillar security for the `password` value
- nessus-agent-formula
  - (Linux) Suppresses gpg verification so the pkg can be installed from a URL

2.12.53 0.17.1

Commit Delta: Change from 0.17.0 release

Released: 2020.01.28

Summary:

- Fixes release date in changelog for 0.17.0
- Removes salt worker special handling for `salt_states` since it is now handled properly in the `Arguments()` class
- pshelp-formula
  - Updates PowerShell help content, including Windows Server 2019

2.12.54 0.17.0

Commit Delta: Change from 0.16.7 release

Released: 2020.01.21

Summary:

- Add support for Windows Server 2019
- Use native markdown processing for PyPI long description
- Deprecate use of 'None' (string) in `config.yaml`
- Add optional `watchmaker_version` node to configuration
- Use Salt 2018.3.4 in default configuration

2.12.55 0.16.7

Commit Delta: Change from 0.16.6 release

Released: 2020.01.06

Summary:

- Pins `PyYAML` dependency when running on Python 3.4 or earlier

2.12.56 0.16.6

Commit Delta: Change from 0.16.5 release

Released: 2019.12.04

Summary:

- Uses CDN URLs for watchmaker config and content, instead of direct S3 URLs
- Pins `backoff` dependency when running on Python 3.4 or earlier

2.12.57 0.16.5

Commit Delta: Change from 0.16.4 release

Released: 2019.09.23

Summary:

- join-domain-formula
  - Add support for restricting Active Directory sites that will be consulted if the `ad_site_name` key-value is set in the pillar
- ash-linux-formula
  - Fix issue with log spamming by `systemd` related to file permissions
- ash-windows-formula
  - Update STIG baselines for 2019-07 SCAP content
- scap-formula
  - Rename DISA content files to reflect SCAP version
  - Update DISA SCAP content to July 2019 release
- salt-content
  - Update SCAP pillar to match filename changes in SCAP formula

2.12.58 0.16.4

Commit Delta: Change from 0.16.3 release

Released: 2019.08.23

Summary:

- Updates documentation on pip usage in Linux to always use `python3 -m pip...`
- dotnet4-formula
  - Adds .NET Framework 4.8 version and associated KB to lookup tables
- fup-formula
  - New salt formula to install packages via URL
- scap-formula
  - (Windows) Adds configuration to allow scan results to be generated when using SCC v5.0.2 and higher
- watchmaker-salt-content

  - (Windows) Adds .NET Framework 4.8 info to dotnet winrepo package content

2.12.59 0.16.3

Commit Delta: Change from 0.16.2 release

Released: 2019.08.7

Summary:

- join-domain-formula
  - (Linux) Modifies method used to retrieve hostname to avoid issues with `hostname -f`
  - (Linux) Improves error messaging if tooling dependencies are not installed
  - (Linux) Modifies domain controller search mechanism to preserve compatibility with EL6
  - (Linux) Logs the computer name in the domain-join output
- mcafee-agent-formula
  - (Linux) Adds a pillar option to pass args to the mcafee agent installer
  - (Linux) Fixes match on OS version to ensure firewall ports are opened
- name-computer-formula
  - (Linux) Updates /etc/hosts with hostname fqdn, when the domain name is provided

2.12.60 0.16.2

Commit Delta: Change from 0.16.1 release

Released: 2019.07.11

Summary:

- join-domain-formula
  - Fixes detection of running system's join state, searches for shortname, and retries joins
  - Improves compatibility with strict Bash
  - Adds option to skip GPG check
- amazon-inspector-formula
  - Adds option to skip GPG check
- splunkforwarder-formula
  - Redirects splunk log folder with symlink
  - Adds option to skip GPG check

2.12.61 0.16.1

Commit Delta: Change from 0.16.0 release

Released: 2019.06.21

Summary:

- join-domain-formula
  - Updates find-collision.sh ldap search to include uppercase and lowercase versions of provided hostname
- scap-formula
  - Adds script to build OSCAP content with 'stig' profile included for CentOS
  - Updates OSCAP content to v0.1.44
- watchmaker-salt-content
  - Switches Linux scap profile pillar settings to 'stig'

2.12.62 0.16.0

Commit Delta: Change from 0.15.2 release

Released: 2019.05.10

Summary:

- Adds salt content locally as a submodule to better support Watchmaker standalone packages
- dotnet4-formula
  - Updates formula to support the use of Python3 versions of Salt
- join-domain-formula
  - Adds additional enhancements and logic to better handle the domin-join process in Linux

2.12.63 0.15.2

Commit Delta: Change from 0.15.1 release

Released: 2019.04.12

Summary:

- ash-linux-formula
  - Removes outdated and conflicting states to allow setting of custom banner text
- join-domain-formula
  - Fixes issue with improper handling of admin names with spaces in Windows

2.12.64 0.15.1

Commit Delta: Change from 0.15.0 release

Released: 2019.04.05

Summary:

- join-domain-formula
  - (Linux) Avoids `unique` jinja filter to preserve compatibility for older versions of salt

2.12.65 0.15.0

Commit Delta: Change from 0.14.2 release

Released: 2019.04.04

Summary:

- Updates documentation to install pip using `ensurepip` module instead of external `get-pip.py`
- ash-linux-formula
  - Adds pillar option to set content for `/etc/issue` login banner
- join-domain-formula
  - (Linux) Adds pillar option to pass a list of domains to add to the trust list

2.12.66 0.14.2

Commit Delta: Change from 0.14.1 release

Released: 2019.03.26

Summary:

- join-domain-formula
  - Corrects regression on Windows to support adding admin groups that have spaces in the name

2.12.67 0.14.1

Commit Delta: Change from 0.14.0 release

Released: 2019.03.18

Summary:

- Fixes Python 2.6 incompatibility introduced by new version of PyYAML
- join-domain-formula
  - Fixes issue adding admin groups/users to Windows systems with recent versions of Salt

2.12.68 0.14.0

Commit Delta: Change from 0.13.0 release

Released: 2019.03.06

Summary:

- Adds additional documentation to answer common EL7 security scan findings
- ash-linux-formula
  - Implements additional Salt states to address security scan issues
    - Capability to manage GRUB password configuration
    - IgnoreRhosts setting in SSH daemon configuration
    - CIS remediation handlers (CIS 5.2.3, CIS 5.2.5)
  - Adds Salt state to update audit-rule changes without a system reboot
- scap-formula
  - Updates SCAP Security Guide content to v0.1.41

2.12.69 0.13.0

Commit Delta: Change from 0.12.1 release

Released: 2019.01.29

Summary:

- amazon-inspector-formula
  - New salt formula distributed with watchmaker
  - Installs amazon-inspector agent
- Refactor watchmaker
  - Change naming mechanism from LinuxManager to LinuxPlatformManager
  - Change naming mechanism from WindowsManager to WindowsPlatformManager
  - Change naming mechanism from Manager to PlatformManager
  - Added abstract class WorkerBase for Workers to inherit from
- ash-linux-formula
  - Change ipv6 check to use if_inet6 file
  - Import correct source of fopen function
  - Configure Postfix to only use ipv4 when ipv6 is disabled

2.12.70 0.12.1

Commit Delta: Change from 0.12.0 release

Released: 2018.12.17

Summary:

- ash-windows-formula
  - Corrects yaml syntax error in win2016 DC baseline

2.12.71 0.12.0

Commit Delta: Change from 0.11.0 release

Released: 2018.12.13

Summary:

- Adds `valid_environments` option to config to allow for the restriction of environment selection

2.12.72 0.11.0

Commit Delta: Change from 0.10.3 release

Released: 2018.11.08

Summary:

- Adds enhancement to ensure `--admin-groups` parameters are lowercase on Linux systems
- Adds additional information to the `--version` flag
- Default values are now shown in help output
- scap-formula
  - Incorporates content from latest DISA SCAP benchmarks
    - Microsoft .Net Framework 4 STIG Benchmark - Ver 1, Rel 5
    - Microsoft Windows 2008 R2 DC STIG Benchmark - Ver 1, Rel 5
    - Microsoft Windows 2008 R2 MS STIG Benchmark - Ver 1, Rel 30
    - Microsoft Windows Server 2016 STIG Benchmark - Ver 1, Rel 31
    - Red Hat 6 STIG Benchmark - Ver 1, Rel 21
    - Red Hat 7 STIG Benchmark - Ver 2, Rel 1

2.12.73 0.10.3

Commit Delta: Change from 0.10.2 release

Released: 2018.10.18

Summary:

- ash-windows-formula
  - Updates Formula to Support Salt 2017.7.x and 2018.3.x
  - Removed admin account rename from delta state

2.12.74 0.10.2

Commit Delta: Change from 0.10.1 release

Released: 2018.09.27

Summary:

- Adds a gitlab-ci pages config to build Watchmaker docs
- Uses new hosting location to retrieve Salt packages
- Restricts click version on py2.6
- ash-windows-forumula
  - New hosting location being used for all packages
- pshelp-formula
  - Removed byte-order-mark unicode character at beginning of init.sls file

2.12.75 0.10.1

Commit Delta: Change from 0.10.0 release

Released: 2018.08.09

Summary:

- No functional changes; just patches the CI/release configuration

2.12.76 0.10.0

Commit Delta: Change from 0.9.6 release

Released: 2018.08.08

Summary:

- Provides standalone packages that bundle the Python runtime together with Watchmaker and its dependencies
  - See https://watchmaker.cloudarmor.io/en/stable/installation.html
- ash-linux-formula
  - (el7) Ensures packages are up-to-date
  - (el7) Ensures firewalld is installed and running

- splunk-forwarder-formula
  - (linux) Uses a symlink to ensure logs are in the /var/log partition
- dotnet4-formula
  - Adds support for .NET 4.7.2
- nessus-agent-formula
  - New salt formula distributed with Watchmaker

2.12.77 0.9.6

Commit Delta: Change from 0.9.5 release

Released: 2018.05.16

Summary:

- windows-update-agent-formula
  - Supports new windows update settings, `AlwaysAutoRebootAtScheduledTime` and `AlwaysAutoRebootAtScheduledTimeMinutes`
- scap-formula
  - Incorporates content from OpenSCAP Security Guide v0.1.39-1

2.12.78 0.9.5

Commit Delta: Change from 0.9.4 release

Released: 2018.04.11

Summary:

- [PR #574] Updates Windows userdata example to execute pip using `python -m` when upgrading pip
- windows-update-agent-formula
  - Uses newer arguments for reg state, `vname` and `vdata`
  - Reduces duplication in windows update data model
  - Nests the windows update pillar options under the standard `lookup` key

2.12.79 0.9.4

Commit Delta: Change from 0.9.3 release

Released: 2018.04.09

Summary:

- ash-windows-formula
  - Updates STIG baselines to address all findings in latest SCAP benchmarks

2.12.80 0.9.3

Commit Delta: Change from 0.9.2 release

Released: 2018.03.08

Summary:

- scap-formula
  - Incorporates content from OpenSCAP Security Guide v0.1.38-1
  - Incorporates content from latest DISA SCAP benchmarks
    - Microsoft Internet Explorer 11 STIG Benchmark - Ver 1, Rel 11
    - Microsoft Windows 10 STIG Benchmark - Ver 1, Rel 10
    - Microsoft Windows 2008 R2 DC STIG Benchmark - Ver 1, Rel 27
    - Microsoft Windows 2008 R2 MS STIG Benchmark - Ver 1, Rel 28
    - Microsoft Windows 2012 and 2012 R2 DC STIG Benchmark - Ver 2, Rel 11
    - Microsoft Windows 2012 and 2012 R2 MS STIG Benchmark - Ver 2, Rel 11
    - Microsoft Windows 8/8.1 STIG Benchmark - Ver 1, Rel 21
    - Microsoft Windows Server 2016 STIG Benchmark - Ver 1, Rel 4
    - Red Hat 6 STIG Benchmark - Ver 1, Rel 18
    - Red Hat 7 STIG Benchmark - Ver 1, Rel 2
- dotnet4-formula
  - Skips dotnet4 hotfix install if a newer version is already installed
  - Creates per-OS maps for hotfix updates, since the hotfix id varies per OS

2.12.81 0.9.2

Commit Delta: Change from 0.9.1 release

Released: 2018.02.20

Summary:

- dotnet4-formula
  - Passes version correctly to module.run

2.12.82 0.9.1

Commit Delta: Change from 0.9.0 release

Released: 2018.02.17

Summary:

- This version was effectively a no-op, as the submodule was not updated as intended
- ~dotnet4-formula~
  - ~Passes version correctly to module.run~

2.12.83 0.9.0

Commit Delta: Change from 0.8.0 release

Released: 2018.02.12

Summary:

- [Issue #499][PR #513] Includes additional details about the platform and python version in the watchmaker log
- [Issue #500][PR #512] Retries file retrieval up to 5 times
- [Issue #501][PR #507] Uses urllib handlers to retrieve all files
  - Deprecates the argument `--s3-source`; to retrieve a file from an S3 bucket use the syntax: `s3://<bucket>/<key>`
  - Local files may be specified as absolute or relative paths, and may or may not be prefixed with `file://`
- [PR #496] Moves CloudFormation and Terraform templates to their own project, terraform-aws-watchmaker
- [PR #491] Improves compatibility of the watchmaker bootstrap.ps1 script when executed by an Azure custom script extension
- [Issue #430][PR #487] Writes watchmaker salt config to a custom path:
  - Windows: `C:\Watchmaker\Salt\conf`
  - Linux: `/opt/watchmaker/salt`
- scap-formula
  - Incorporates content from OpenSCAP Security Guide v0.1.37-1

2.12.84 0.8.0

Commit Delta: Change from 0.7.2 release

Released: 2018.01.02

Summary:

- [Issue #415][PR #458] Forwards watchmaker log entries from the Windows Event Log to the EC2 System Log (Windows-only)
- [PR #425] Adds a log handler that writes watchmaker log entries to the Windows Event Log (Windows-only)
- [Issue #434][PR #457] Updates doc build to replace `recommonmark` functionality entirely with `m2r`
- [PR #437] Modfies CloudFormation templates to use aws cli utility to retrieve the appscript rather than use the functionality built-in to the cfn bootstrap
- [PR #467] Sets environment variables for aws cli when executing the appscript option in the watchmaker CloudFormation templates

2.12.85 0.7.2

Commit Delta: Change from 0.7.1 release

Released: 2017.12.13

Summary:

- Installs `futures` only on Python 2 – no functional changes

2.12.86 0.7.1

Commit Delta: Change from 0.7.0 release

Released: 2017.12.04

Summary:

- Fixes readthedocs build – no functional changes

2.12.87 0.7.0

Commit Delta: Change from 0.6.6 release

Released: 2017.11.21

Summary:

- [PR #409] Provides terraform modules that deploy the watchmaker CloudFormation templates
- [Issue #418][PR #419] Adds an `exclude-states` argument to the SaltWorker; specified states will be excluded from the salt state execution
- ash-windows-formula
  - Incorporates security settings from the DISA October quarterly release
- join-domain-formula
  - (Windows) Adds WMI method to set DNS search suffix
  - (Windows) Tests for the EC2Config XML settings file before modifying it
- scap-formula
  - (Linux) Distributes scap content from SCAP Security Guide v0.1.36-1
  - Distributes scap content from the DISA October quarterly release
- splunkforwarder-formula
  - Supports configuration of splunk log sources from pillar and grains inputs

2.12.88 0.6.6

Commit Delta: Change from 0.6.5 release

Released: 2017.10.18

Summary:

- ash-linux-formula
  - (el7) Fixes typos in the firewalld "safety" scripts that resulted in a failure when firewalld was reloaded
- mcafee-agent-formula
  - (el7) Adds required inbound ports to all firewalld zones, to support the event where the default zone is modified from "public"
- splunkforwarder-formula
  - (el7) Adds required outbound ports to the OUTPUT chain; previously, they were mistakenly being added as inbound rules

2.12.89 0.6.5

Commit Delta: Change from 0.6.4 release

Released: 2017.09.29

Summary:

- [PR #391] Updates CloudFormation templates with a parameter that exposes the option to use the S3 API and the instance role to retrieve the Watchmaker content archive
- ash-linux-formula
  - (el7) Updates firewalld "safety" state so that firewalld remains in the active state; the prior approach left firewalld dead/inactive, until the service was restarted or the system was rebooted

2.12.90 0.6.4

Commit Delta: Change from 0.6.3 release

Released: 2017.09.22

Summary:

- [PR #381] Restricts `wheel` version on Python 2.6 to be less than or equal to 0.29.0, as `wheel` 0.30.0 removed support for py26.

2.12.91 0.6.3

Commit Delta: Change from 0.6.2 release

Released: 2017.08.11

Summary:

- ash-linux-formula
  - (el7) Includes a "safety" state for firewalld that ensures SSH inbound access will remain available, in the event the default zone is set to "drop"

2.12.92 0.6.2

Commit Delta: Change from 0.6.1 release

Released: 2017.08.07

Summary:

- ash-linux-formula
  - (el6) Improve the method of disabling the sysctl option `ip_forward`, to account for the behavior of the `aws-vpc-nat` rpm
- scap-formula
  - (elX) Updates openscap security guide content to version 0.1.34-1

2.12.93 0.6.1

Commit Delta: Change from 0.6.0 release

Released: 2017.08.01

Summary:

- ash-linux-formula
  - Modified the FIPS custom execution module to discover the boot partition and add the `boot=` line to the grub configuration

2.12.94 0.6.0

Commit Delta: Change from 0.5.1 release

Released: 2017.07.25

Summary:

- ash-linux-formula
  - Updates the EL7 stig baseline to manage the FIPS state. The state defaults to `enabled` but can be overridden via a pillar or grain, `ash-linux:lookup:fips-state`. The grain takes precedence over the pillar. Valid values are `enabled` or `disabled`
- ash-windows-formula
  - Updates the STIG baselines for Windows Server 2016 member servers and domain controllers with SCAP content from the DISA v1r1 SCAP benchmark release
- join-domain-formula
  - Fixes an issue when joining Windows 2016 servers to a domain, where the Set-DnsSearchSuffix.ps1 helper would fail because the builtin PowerShell version does not work when `$null` is used in a ValidateSet. The equivalent value must now be passed as the string, `"null"`
- scap-formula
  - Adds SCAP content for the Window Server 2016 SCAP v1r1 Benchmark

2.12.95 0.5.1

Commit Delta: Change from 0.5.0 release

Released: 2017.07.08

Summary:

- [Issue #341][PR #342] Manages selinux around salt state execution. In some non-interactive execution scenarios, if selinux is enforcing it can interfere with the execution of privileged commands (that otherwise work fine when executed interactively). Watchmaker now detects if selinux is enforcing and temporarily sets it to permissive for the duration of the salt state execution

2.12.96 0.5.0

Commit Delta: Change from 0.4.4 release

Released: 2017.06.27

Summary:

- [Issue #331][PR #332] Writes the `role` grain to the key expected by the ash-windows formula. Fixes usage of the `--ash-role` option in the salt worker
- [Issue #329][PR #330] Outputs watchmaker version at the debug log level
- [Issue #322][PR #323][PR #324] Fixes py2/py3 compatibility bug in how the yum worker handles file opening to check the Linux distro
- [Issue #316][PR #320] Improves logging when salt state execution fails due to failed a state. The salt output is now returned to the salt worker, which processes the output, identifies the failed state, and raises an exception with the state failure
- join-domain-formula
  - (Linux) Reworks the pbis config states to make the logged output more readable

2.12.97 0.4.4

Commit Delta: Change from 0.4.3 release

Released: 2017.05.30

Summary:

- join-domain-formula
  - (Linux) Ignores a bad exit code from pbis config utility. The utility will return exit code 5 when modifying the NssEnumerationEnabled setting, but still sets the requested value. This exit code is now ignored

2.12.98 0.4.3

Commit Delta: Change from 0.4.2 release

Released: 2017.05.25

Summary:

- name-computer-formula
  - (Linux) Uses an alternate method of working around a bad code-path in salt that does not handle quoted values in /etc/sysconfig/network.

2.12.99 0.4.2

Commit Delta: Change from 0.4.1 release

Released: 2017.05.19

Summary:

- [PR #301] Sets the grains for admin_groups and admin_users so the keys are named as expected by the join-domain formula
- ash-linux-formula
  - Adds a custom module that lists users from the shadow file
  - Gets local users from the shadow file rather than `user.list_users`. Prevents a domain-joined system from attempting to iterate over all domain users (and potentially deadlocking on especially large domains)
- join-domain-formula
  - Modifies PBIS install method to use RPMs directly, rather than the SHAR installer
  - Updates approaches to checking for collisions and current join status to better handle various scenarios: not joined, no collision; not joined, collision; joined, computer object present; joined, computer object missing
  - Disables NSS enumeration to prevent PBIS from querying user info from the domain for every call to getent (or equivalents); domain-based user authentication still works fine
- name-computer-formula
  - (Linux) Does not attempt to retain network settings, to avoid a bug in salt; will be revisited when a patched salt version has been released

2.12.100 0.4.1

Commit Delta: Change from 0.4.0 release

Released: 2017.05.09

Summary:

- (EL7) Running *watchmaker* against EL7 systems will now pin the resulting configuration to the watchmaker version. See the updates to the two formulas in this version. Previously, *ash-linux* always used the content from the `scap-security-guide` rpm, which was updated out-of-sync with *watchmaker*, and so the resulting configuration could not be pinned by pinning the *watchmaker* version. With this version, *ash-linux* uses content distributed by *watchmaker*, via *scap-formula*, and so the resulting configuration will always be same on EL7 for a given version of *watchmaker* (as has always been the case for the other supported operating systems).
- ash-linux-formula

  - Supports getting scap content locations from pillar
- scap-formula
  - Updates stig content with latest benchmark versions
  - Adds openscap ds.xml content, used to support remediate actions

2.12.101 0.4.0

Commit Delta: Change from 0.3.1 release

Released: 2017.05.06

Summary:

- [PR #286] Sets the computername grain with the correct key expected by the formula
- [PR #284] Converts cli argument parsing from `argparse` to `click`. This modifies the `watchmaker` depedencies, which warranted a 0.x.0 version bump. Cli and API arguments remain the same, so the change should be backwards-compatible.
- name-computer-formula
  - Adds support for getting the computername from pillar
  - Adds support for validating the specified computername against a pattern
- pshelp-formula
  - Attempts to address occasional stack overflow exception when updating powershell help

2.12.102 0.3.1

Commit Delta: Change from 0.3.0 release

Released: 2017.05.01

Summary:

- [PR #280] Modifies the dynamic import of boto3 to use only absolute imports, as the previous approach (attempt absolute and relative import) was deprecated in Python 3.3
- ntp-client-windows-formula:
  - Stops using deprecated arguments on reg.present states, which cleans up extraneous log messages in watchmaker runs under some configurations
- join-domain-formula:
  - (Windows) Sets the DNS search suffix when joining the domain, including a new pillar config option, `ec2config` to enable/disable the EC2Config option that also modifies the DNS suffix list.

2.12.103 0.3.0

Commit Delta: Change from 0.2.4 release

Released: 2017.04.24

Summary:

- [Issue #270] Defaults to a platform-specific log directory when call from the CLI:
  - Windows: `${Env:SystemDrive}\Watchmaker\Logs`
  - Linux: `/var/log/watchmaker`
- [PR #271] Modifies CLI arguments to use explicit log-levels rather than a verbosity count. Arguments have been adjusted to better accommodate the semantics of this approach:
  - Uses `-l|--log-level` instead of `-v|--verbose`
  - `-v` and `-V` are now both used for `--version`
  - `-d` is now used for `--log-dir`

2.12.104 0.2.4

Commit Delta: Change from 0.2.3 release

Released: 2017.04.20

Summary:

- Fixes a bad version string

2.12.105 0.2.3

Commit Delta: Change from 0.2.2 release

Released: 2017.04.20

Summary:

- [Issue #262] Merges lists in pillar files, rather than overwriting them
- [Issue #261] Manages the enabled/disabled state of the salt-minion service, before and after the install
- splunkforwarder-formula
  - (Windows) Ignores false bad exits from Splunk clone-prep-clear-config

2.12.106 0.2.2

Commit Delta: Change from 0.2.1 release

Released: 2017.04.15

Summary:

- [PR #251] Adds CloudFormation templates that integrate Watchmaker with an EC2 instance or Autoscale Group
- join-domain-formula
  - (Linux) Corrects tests that determine whether the instance is already joined to the domain

2.12.107 0.2.1

Commit Delta: Change from 0.2.0 release

Released: 2017.04.10

Summary:

- ash-linux-formula
  - Reduces spurious stderr output
  - Removes notify script flagged by McAfee scans
- splunkforwarder-formula
  - (Windows) Clears system name entries from local Splunk config files

2.12.108 0.2.0

Commit Delta: Change from 0.1.7 release

Released: 2017.04.06

Summary:

- [Issue #238] Captures all unhandled exceptions and logs them
- [Issue #234] Stops the salt service prior to managing salt formulas, to ensure that the filesystem does not throw any errors about the files being locked
- [Issue #72] Manages salt service so the service state after watchmaker completes is the same as it was prior to running watchmaker. If the service was running beforehand, it remains running afterwards. If the service was stopped (or non-existent) beforehad, the service remains stopped afterwards
- [Issue #163] Modifies the `user_formulas` config option to support a map of `<formula_name>:<formula_url>`
- [PR #235] Extracts salt content to the same target `srv` location for both Window and Linux. Previously, the salt content was extracted to different points in the filesystem hierarchy, which required different content for Windows and Linux. Now the same salt content archive can be used for both
- [PR #242] Renames salt worker param `content_source` to `salt_content`
- systemprep-formula
  - Deprecated and removed. Replaced by new salt content structure that uses native salt capabilities to map states to a system
- scc-formula
  - Deprecated and removed. Replaced by scap-formula
- scap-formula
  - New bundled salt formula. Provides SCAP scans using either `openscap` or `scc`
- pshelp-formula
  - New bundled salt formula. Installs updated PowerShell help content to Windows systems

2.12.109 0.1.7

Commit Delta: Change from 0.1.6 release

Released: 2017.03.23

Summary:

- Uses threads to stream stdout and stderr to the watchmaker log when executing a command via subproces
- [Issue #226] Minimizes salt output of successful states, to make it easier to identify failed states
- join-domain-formula
  - (Linux) Exits with stateful failure on a bad decryption error
- mcafee-agent-formula
  - (Linux) Avoids attempting to diff a binary file
  - (Linux) Installs `ed` as a dependency of the McAfee VSEL agent
- scc-formula
  - Retries scan up to 5 times if scc exits with an error

2.12.110 0.1.6

Commit Delta: Change from 0.1.5 release

Released: 2017.03.16

Summary:

- ash-linux-formula
  - Provides same baseline states for both EL6 and EL7

2.12.111 0.1.5

Commit Delta: Change from 0.1.4 release

Released: 2017.03.15

Summary:

- ash-linux-formula
  - Adds policies to disable insecure Ciphers and MACs in sshd_config
- ash-windows-formula
  - Adds `scm` and `stig` baselines for Windows 10
  - Adds `scm` baseline for Windows Server 2016 (Alpha)
  - Updates all `scm` and `stig` baselines with latest content
- mcafee-agent-formula
  - Uses firewalld on EL7 rather than iptables
- scc-formula
  - Skips verification of GPG key when install SCC RPM

- splunkforwarder-formula
  - Uses firewalld on EL7 rather than iptables

2.12.112 0.1.4

Commit Delta: Change from 0.1.3 release

Released: 2017.03.09

Summary:

- [Issue #180] Fixes bug where file_roots did not contain formula paths

2.12.113 0.1.3

Commit Delta: Change from 0.1.2 release

Released: 2017.03.08

Summary:

- [Issue #164] Aligns cli syntax for extra_arguments with other cli opts
- [Issue #165] Removes ash_role from default config file
- [Issue #173] Fixes exception when re-running watchmaker

2.12.114 0.1.2

Commit Delta: Change from 0.1.1 release

Released: 2017.03.07

Summary:

- Adds a FAQ page to the docs
- Moves salt formulas to the correct location on the local filesystem
- join-domain-formula:
  - (Linux) Modifies decryption routine for FIPS compliance
- ash-linux-formula:
  - Removes several error exits in favor of warnings
  - (EL7-alpha) Various patches to improve support for EL7
- dotnet4-formula:
  - Adds support for .NET 4.6.2
  - Adds support for Windows Server 2016
- emet-formula:
  - Adds support for EMET 5.52

2.12.115 0.1.1

Commit Delta: Change from 0.1.0 release

Released: 2017.02.28

Summary:

- Adds more logging messages when downloading files

2.12.116 0.1.0

Commit Delta: N/A

Released: 2017.02.22

Summary:

- Initial release!

CHAPTER

THREE

SUPPORTED OPERATING SYSTEMS

- Enterprise Linux 8 (RHEL/CentOS Stream/Oracle Linux)
- Enterprise Linux 7 (RHEL/CentOS)
- Windows Server 2019
- Windows Server 2016
- Windows Server 2012 R2
- Windows 10
- Windows 8.1

CHAPTER

FOUR

SUPPORTED PYTHON VERSIONS

- Python 3.6 and later
- Python 2.7 and later

CHAPTER

FIVE

SUPPORTED SALT VERSIONS

- Salt 2018.3, from 2018.3.4 and later
- Salt 2019.2, from 2019.2.5 and later
- Salt 300x, from 3003 and later

PYTHON MODULE INDEX

W

INDEX

W

Y