Watchmaker Documentation

Plus3 IT Systems

Feb 27, 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.
- `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 be passed to pip when installing package(s). Options with values should be passed with the option and value as separate list items.

  Linux example where emoji is a value that corresponds to the `--progress-bar` option:

```
linux:
  - salt:
      pip_args:
        - --ignore-installed
        - --progress-bar=emoji
```

- `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,
        # 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:
        #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.
```

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
```

(continues on next page)

(continued from previous page)

```
# 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/3.8.10/python-3.8.10-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
```

(continues on next page)

(continued from previous page)

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

2.4.1 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	RHEL-07-040530 SV-86899

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.

2.5 Supported SCAP Benchmarks

2.5.1 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

2.5.2 Linux

- Red Hat Enterprise Linux STIG Benchmark (EL7) (EL8 work in progress)
- OpenSCAP Security Guide (EL7) (EL8 work in progress)

  New benchmark versions are incorporated as they are released

2.6 Frequently Asked Questions

2.6.1 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`.

2.6.2 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.

2.6.3 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.

2.6.4 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.

2.6.5 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.

2.6.6 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.

2.6.7 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
```

2.6.8 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
```

2.6.9 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.

2.6.10 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"
```

2.6.11 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.

2.7 API Reference

2.7.1 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.

2.8 Contributing

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

2.8.1 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.

2.8.2 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.

2.8.3 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. :)

2.8.4 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
```

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.

2.8.5 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.

2.8.6 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
```

2.8.7 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
```

(continues on next page)

(continued from previous page)

```
# 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/3.8.10/python-3.8.10-amd64.exe"
$GitUrl = "https://github.com/git-for-windows/git/releases/download/v2.37.1.windows.
↪1/Git-2.37.1-64-bit.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 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
```

(continues on next page)

(continued from previous page)

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

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

2.9 Changelog

2.9.1 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

2.9.2 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

2.9.3 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

2.9.4 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

2.9.5 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

2.9.6 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

2.9.7 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

2.9.8 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

2.9.9 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

2.9.10 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

2.9.11 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

2.9.12 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)

2.9.13 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"}'`

2.9.14 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"}'`

2.9.15 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

2.9.16 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

2.9.17 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

2.9.18 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

2.9.19 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.

2.9.20 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.

2.9.21 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.

2.9.22 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

2.9.23 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

2.9.24 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

2.9.25 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

2.9.26 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

2.9.27 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

2.9.28 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

2.9.29 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

2.9.30 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

2.9.31 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)

2.9.32 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

2.9.33 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

2.9.34 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

2.9.35 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

2.9.36 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

2.9.37 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

2.9.38 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

2.9.39 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

2.9.40 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

2.9.41 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

2.9.42 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

2.9.43 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

2.9.44 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

2.9.45 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

2.9.46 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

2.9.47 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

2.9.48 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

2.9.49 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

2.9.50 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'

2.9.51 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

2.9.52 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

2.9.53 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

2.9.54 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

2.9.55 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

2.9.56 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

2.9.57 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

2.9.58 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

2.9.59 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

2.9.60 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

2.9.61 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

2.9.62 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

2.9.63 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

2.9.64 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

2.9.65 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

2.9.66 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

2.9.67 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

2.9.68 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

2.9.69 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

2.9.70 0.9.2

Commit Delta: Change from 0.9.1 release

Released: 2018.02.20

Summary:

- dotnet4-formula
  - Passes version correctly to module.run

2.9.71 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~

2.9.72 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

2.9.73 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

2.9.74 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

2.9.75 0.7.1

Commit Delta: Change from 0.7.0 release

Released: 2017.12.04

Summary:

- Fixes readthedocs build – no functional changes

2.9.76 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

2.9.77 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

2.9.78 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

2.9.79 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.

2.9.80 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"

2.9.81 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

2.9.82 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

2.9.83 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

2.9.84 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

2.9.85 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

2.9.86 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

2.9.87 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.

2.9.88 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

2.9.89 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

2.9.90 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

2.9.91 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.

2.9.92 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`

2.9.93 0.2.4

Commit Delta: Change from 0.2.3 release

Released: 2017.04.20

Summary:

- Fixes a bad version string

2.9.94 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

2.9.95 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

2.9.96 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

2.9.97 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

2.9.98 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

2.9.99 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

2.9.100 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

2.9.101 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

2.9.102 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

2.9.103 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

2.9.104 0.1.1

Commit Delta: Change from 0.1.0 release

Released: 2017.02.28

Summary:

- Adds more logging messages when downloading files

2.9.105 0.1.0

Commit Delta: N/A

Released: 2017.02.22

Summary:

- Initial release!

CHAPTER

THREE

SUPPORTED OPERATING SYSTEMS

- Enterprise Linux 7 (RHEL/CentOS/etc)
- 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