

---

# **Watchmaker Documentation**

***Release***

**Plus3 IT Systems**

**Jan 02, 2018**



---

## Contents

---

<b>1 Overview</b>	<b>3</b>
<b>2 Contents</b>	<b>5</b>
<b>3 Supported Operating Systems</b>	<b>37</b>
<b>4 Supported Python Versions</b>	<b>39</b>
<b>Python Module Index</b>	<b>41</b>



Applied Configuration Management

---



# CHAPTER 1

---

## 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.



## 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.

```
pip install watchmaker
```

If you do not have Python or `pip`, this [Python installation guide](#) can guide you through the process.

### 2.1.2 From sources

Watchmaker can also be built and installed from source, using `git` and `pip`. The sources 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:

```
pip install .
```

## 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 Config.yaml Parent Nodes

#### **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.

### 2.2.2 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.

- `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"
```

- `salt_states` (*string, comma-separated*): User-defined salt states to apply.

```
salt_states: foo,bar
```

- `exclude_states` (*string, comma-separated*): States to exclude from execution of salt states.
- `s3_source` (*boolean*): Use S3 utilities to retrieve content instead of http(s) utilities. For S3 utilities to work, the system must have boto credentials configured that allow access to the S3 bucket.
- `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.
- `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 specified, 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

```

### 2.2.3 Example config.yaml

```

all:
  - salt:
      admin_groups: None
      admin_users: None
      computer_name: None
      environment: False
      ou_path: None
      salt_content: https://s3.amazonaws.com/watchmaker/salt-content.zip
      salt_states: Highstate
      s3_source: False
      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
        #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://s3.amazonaws.com/systemprep-repo/linux/saltstack/salt/yum.
↪repos/salt-reposync-el6.repo
          - dist: amazon
            el_version: 6
            url: https://s3.amazonaws.com/systemprep-repo/linux/saltstack/salt/yum.
↪repos/salt-reposync-amzn.repo
        #SaltEL7:
          - dist:
              - redhat
              - centos
            el_version: 7
            url: https://s3.amazonaws.com/systemprep-repo/linux/saltstack/salt/yum.
↪repos/salt-reposync-el7.repo
      - salt:
          salt_debug_log: None
          install_method: yum
          bootstrap_source: None

```

```

git_repo: None
salt_version: None

windows:
- salt:
  salt_debug_log: None
  installer_url: https://s3.amazonaws.com/systemprep-repo/windows/salt/Salt-
↳Minion-2016.11.6-AMD64-Setup.exe

```

## 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.

```

# 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.
  --s3-source              Use S3 utilities to retrieve content instead
                           of http/s utilities. Boto3 must be
                           installed, and boto3 credentials must be
                           configured that allow access to the S3
                           bucket.
  -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.

```

<code>-e, --env TEXT</code>	Set a salt grain that specifies the environment in which the system is being built. E.g. dev, test, or prod
<code>-p, --ou-path TEXT</code>	Set 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"
<code>--help</code>	Show this message and exit.

### 2.3.2 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
PIP_URL=https://bootstrap.pypa.io/get-pip.py
PYPI_URL=https://pypi.org/simple

# Install pip
curl "$PIP_URL" | python - --index-url="$PYPI_URL" wheel==0.29.0

# Install watchmaker
pip install --index-url="$PYPI_URL" --upgrade pip setuptools 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:
- |
  PIP_URL=https://bootstrap.pypa.io/get-pip.py
  PYPI_URL=https://pypi.org/simple

  # Install pip
  curl "$PIP_URL" | python - --index-url="$PYPI_URL" wheel==0.29.0

  # Install watchmaker
```

```
pip install --index-url="$PYPI_URL" --upgrade pip setuptools 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://raw.githubusercontent.com/plus3it/watchmaker/master/docs/
↳ files/bootstrap/watchmaker-bootstrap.ps1"
$PythonUrl = "https://www.python.org/ftp/python/3.6.3/python-3.6.3-amd64.exe"
$PypiUrl = "https://pypi.org/simple"

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

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

# Install watchmaker
pip install --index-url="$PypiUrl" --upgrade pip setuptools 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.

---

**Note:** Note that the links in this section are intended for viewing the templates in a web browser. See the *Direct Downloads* section for links to the raw files.

---

## Cloudformation templates

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

### Cloudformation parameter-maps

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 templates above.

- [Linux Autoscale Params](#)
- [Linux Instance Params](#)
- [Windows Autoscale Params](#)
- [Windows Instance Params](#)

### watchmaker as a Terraform template

Watchmaker can be integrated into a Terraform template as well. By wrapping the example CloudFormation templates within their respective Terraform template they become deployable and manageable from within the [Terraform cli](#). These templates are intended as examples for you to modify and extend as you need.

---

#### Note:

- These templates assume that the accompanying CloudFormation template is in the same directory.
  - The links in this section are intended for viewing the templates in a web browser. See the [Direct Downloads](#) section for links to the raw files.
- 

Variable values can be input interactively via the terraform console or via a Terraform module. An example Terraform module that calls the lx-autoscale template is shown below.

```
module "test-lx-instance" {
  source = "github.com/plus3it/watchmaker//docs/files/templates/lx-instance"

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

### Terraform templates

- [Linux Autoscale Group](#)
- [Linux Instance](#)
- [Windows Autoscale Group](#)
- [Windows Instance](#)

## 2.3.3 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

```
{
  "fileUri": ["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](#).

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:

```
{
  "fileUri": ["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](#).

### 2.3.4 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'
arguments.s3_source = False

client = watchmaker.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](#).

### 2.3.5 Direct downloads

The following links can be used for directly fetching (e.g., via `curl`, `wget`, etc.) resources previously noted on this page:

CFN Template Files	CFN Parameter Files	TF Template Files
<a href="#">Linux AutoScale</a>	<a href="#">Linux Autoscale</a>	<a href="#">Linux AutoScale</a>
<a href="#">Linux Instance</a>	<a href="#">Linux Instance</a>	<a href="#">Linux Instance</a>
<a href="#">Windows Autoscale</a>	<a href="#">Windows Autoscale</a>	<a href="#">Windows Autoscale</a>
<a href="#">Windows Instance</a>	<a href="#">Windows Instance</a>	<a href="#">Windows Instance</a>

## 2.4 Frequently Asked Questions

### 2.4.1 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`.

### 2.4.2 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.

### 2.4.3 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.

### 2.4.4 How do I know if watchmaker has completed without errors?

By default, watchmaker will reboot the system after a successful 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.

### 2.4.5 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.

### 2.4.6 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.

## 2.4.7 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 --local ash.fips_disable`
- Enable FIPS-mode: `salt-call --local ash.fips_enable`

And then rebooting the system.

## 2.5 API Reference

### 2.5.1 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`. Additionally, 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'` (the string) will not apply any salt states. A value of `'Highstate'` will apply the salt highstate. (*Default: None*)

- **s3\_source** – (*bool*) Use S3 utilities to retrieve content instead of http/s utilities. For S3 utilities to work, `boto3` must be installed, and the system must have boto credentials configured that allow access to the S3 bucket. (*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.base**

Watchmaker base manager.

**class** `watchmaker.managers.base.ManagerBase` (*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.

**download\_file** (*url*, *filename*, *sourceiss3bucket=False*)

Download a file from a web server or S3 bucket.

**Parameters**

- **url** – (`str`) URL to a file.
- **filename** – (`str`) Path where the file will be saved.
- **sourceiss3bucket** – (`bool`) Switch to indicate that the download should use boto3 to download the file from an S3 bucket. (*Default: False*)

**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=u'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.base.LinuxManager` (*system\_params*, *\*args*, *\*\*kwargs*)

Bases: `watchmaker.managers.base.ManagerBase`

Base class for Linux Managers.

Serves as a foundational class to keep OS consistency.

**class** `watchmaker.managers.base.WindowsManager` (*system\_params*, *\*args*, *\*\*kwargs*)

Bases: `watchmaker.managers.base.ManagerBase`

Base class for Windows Managers.

Serves as a foundational class to keep OS consistency.

**class** `watchmaker.managers.base.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.

## watchmaker.managers.workers

Watchmaker workers manager.

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

Bases: *watchmaker.managers.base.WorkersManagerBase*

Manage the worker cadence for Linux systems.

```
worker_cadence()
    Manage worker cadence.
```

```
cleanup()
    Execute cleanup function.
```

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

Bases: *watchmaker.managers.base.WorkersManagerBase*

Manage the worker cadence for Windows systems.

```
worker_cadence()
    Manage worker cadence.
```

```
cleanup()
    Execute cleanup function.
```

## watchmaker.workers

Watchmaker workers module.

### watchmaker.workers.salt

Watchmaker salt worker.

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

Bases: *watchmaker.managers.base.ManagerBase*

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: ''*)
- **s3\_source** – (*bool*) Use S3 utilities to download salt content and user formulas from an S3 bucket. If True, you must also install *boto3* and *botocore*. Those dependencies will not be installed by Watchmaker. (*Default: False*)
- **salt\_content** – (*str*) URL to a salt content archive (zip file) that will be uncompressed in the salt “srv” directory. This typically is used to create a top.sls file and to populate salt’s file\_roots. (*Default: ''*)
  - *Linux*: /srv/salt
  - *Windows*: C:\Salt\srv
- **salt\_states** – (*str*) Comma-separated string of salt states to execute. Accepts two special keywords (case-insensitive). (*Default: ''*)

- `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*: `' '`)

**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. 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: `watchmaker.workers.salt.SaltBase`, `watchmaker.managers.base.LinuxManager`

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: `watchmaker.workers.salt.SaltBase`, `watchmaker.managers.base.WindowsManager`

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: *watchmaker.managers.base.LinuxManager*

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.

**install()**

Install yum repos defined in config file.

## 2.6 Contributing

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

### 2.6.1 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.

### 2.6.2 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.

### 2.6.3 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. :)

### 2.6.4 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.

## 2.6.5 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.

## 2.6.6 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
```

## 2.6.7 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>

PIP_URL=https://bootstrap.pypa.io/get-pip.py
PYPI_URL=https://pypi.org/simple

# Install pip
curl "$PIP_URL" | python - --index-url="$PYPI_URL" wheel==0.29.0
```

```

# Install git
yum -y install git

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

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

# Install watchmaker
cd watchmaker
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://raw.githubusercontent.com/plus3it/watchmaker/master/docs/
↳files/bootstrap/watchmaker-bootstrap.ps1"
$PythonUrl = "https://www.python.org/ftp/python/3.6.3/python-3.6.3-amd64.exe"
$GitUrl = "https://github.com/git-for-windows/git/releases/download/v2.14.3.
↳windows.1/Git-2.14.3-64-bit.exe"
$PypiUrl = "https://pypi.org/simple"

# 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
pip install --index-url="$PypiUrl" --upgrade pip setuptools

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

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

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

```

## 2.7 Changelog

### 2.7.1 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`
- [\[PR #437\]](#) Modifies CloudFormation templates to use `aws cli` utility to retrieve the `appscript` rather than use the functionality built-in to the `cf` bootstrap
- [\[PR #467\]](#) Sets environment variables for `aws cli` when executing the `appscript` option in the watchmaker CloudFormation templates

### 2.7.2 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

### 2.7.3 0.7.1

**Commit Delta:** [Change from 0.7.0 release](#)

**Released:** 2017.12.04

**Summary:**

- Fixes `readthedocs` build – no functional changes

### 2.7.4 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

### 2.7.5 0.6.6

**Commit Delta:** [Change from 0.6.5 release](#)

**Released:** 2017.10.18

**Summary:**

- ash-linux-formula
  - (e17) Fixes typos in the firewalld “safety” scripts that resulted in a failure when firewalld was reloaded
- mcafee-agent-formula
  - (e17) Adds required inbound ports to all firewalld zones, to support the event where the default zone is modified from “public”
- splunkforwarder-formula
  - (e17) Adds required outbound ports to the OUTPUT chain; previously, they were mistakenly being added as inbound rules

### 2.7.6 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
  - (e17) 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

### 2.7.7 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`.

### 2.7.8 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”

### 2.7.9 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

### 2.7.10 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

### 2.7.11 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

### 2.7.12 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

### 2.7.13 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

### 2.7.14 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

### 2.7.15 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`.

### 2.7.16 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

### 2.7.17 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

## 2.7.18 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` dependencies, 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

## 2.7.19 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.

## 2.7.20 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`

## 2.7.21 0.2.4

**Commit Delta:** Change from 0.2.3 release

**Released:** 2017.04.20

**Summary:**

- Fixes a bad version string

## 2.7.22 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`

## 2.7.23 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

## 2.7.24 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

## 2.7.25 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) beforehand, 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

## 2.7.26 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 subprocess
- [\[Issue #226\]](#) 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

## 2.7.27 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

## 2.7.28 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

## 2.7.29 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

## 2.7.30 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

## 2.7.31 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

### 2.7.32 0.1.1

**Commit Delta:** Change from 0.1.0 release

**Released:** 2017.02.28

**Summary:**

- Adds more logging messages when downloading files

### 2.7.33 0.1.0

**Commit Delta:** N/A

**Released:** 2017.02.22

**Summary:**

- Initial release!

## CHAPTER 3

---

### Supported Operating Systems

---

- Enterprise Linux 7 (RHEL/CentOS/etc)
- Enterprise Linux 6 (RHEL/CentOS/etc)
- Windows Server 2016
- Windows Server 2012 R2
- Windows Server 2008 R2
- Windows 10
- Windows 8.1



## CHAPTER 4

---

### Supported Python Versions

---

- Python 3.4 and later
- Python 2.6 and later



**W**

watchmaker, 15  
watchmaker.managers, 17  
watchmaker.managers.base, 17  
watchmaker.managers.workers, 19  
watchmaker.workers, 19  
watchmaker.workers.salt, 19  
watchmaker.workers.yum, 22



**A**

Arguments (class in watchmaker), 15

**C**

call\_process() (watchmaker.managers.base.ManagerBase method), 17

cleanup() (watchmaker.managers.base.ManagerBase method), 18

cleanup() (watchmaker.managers.workers.LinuxWorkersManager method), 19

cleanup() (watchmaker.managers.workers.WindowsWorkersManager method), 19

Client (class in watchmaker), 16

create\_working\_dir() (watchmaker.managers.base.ManagerBase method), 17

**D**

download\_file() (watchmaker.managers.base.ManagerBase method), 17

**E**

extract\_contents() (watchmaker.managers.base.ManagerBase method), 18

**G**

get\_dist\_info() (watchmaker.workers.yum.Yum method), 22

**I**

install() (watchmaker.Client method), 16

install() (watchmaker.workers.salt.SaltLinux method), 21

install() (watchmaker.workers.salt.SaltWindows method), 22

install() (watchmaker.workers.yum.Yum method), 22

**L**

LinuxManager (class in watchmaker.managers.base), 18

LinuxWorkersManager (class in watchmaker.managers.workers), 19

**M**

ManagerBase (class in watchmaker.managers.base), 17

**P**

process\_grains() (watchmaker.workers.salt.SaltBase method), 21

process\_states() (watchmaker.workers.salt.SaltBase method), 21

**R**

run\_salt() (watchmaker.workers.salt.SaltBase method), 20

**S**

SaltBase (class in watchmaker.workers.salt), 19

SaltLinux (class in watchmaker.workers.salt), 21

SaltWindows (class in watchmaker.workers.salt), 21

service\_disable() (watchmaker.workers.salt.SaltBase method), 21

service\_enable() (watchmaker.workers.salt.SaltBase method), 21

service\_start() (watchmaker.workers.salt.SaltBase method), 20

service\_status() (watchmaker.workers.salt.SaltBase method), 20

service\_stop() (watchmaker.workers.salt.SaltBase method), 20

**W**

watchmaker (module), 15

watchmaker.managers (module), 17

watchmaker.managers.base (module), 17

watchmaker.managers.workers (module), 19

watchmaker.workers (module), 19

watchmaker.workers.salt (module), 19

watchmaker.workers.yum (module), 22

WindowsManager (class in watchmaker.managers.base),  
18

WindowsWorkersManager (class in watch-  
maker.managers.workers), 19

worker\_cadence() (watch-  
maker.managers.workers.LinuxWorkersManager  
method), 19

worker\_cadence() (watch-  
maker.managers.workers.WindowsWorkersManager  
method), 19

WorkersManagerBase (class in watch-  
maker.managers.base), 18

## Y

Yum (class in watchmaker.workers.yum), 22