Whitepaper

IT Infrastructure Management

Content Lifecycle Management

SUSE Manager 4 is a best-in-class, open source infrastructure management solution that lowers costs, enhances availability and reduces complexity for the lifecycle management of Linux systems in large, complex and dynamic IT landscapes. You can use SUSE Manager to configure, deploy and administer thousands of Linux systems running on hypervisors; as containers; and on bare metal systems, IoT devices and third-party cloud platforms. You can also use SUSE Manager to manage virtual machines.

Content Lifecycle Management

What is Content Lifecycle Management?

The concept of Content Lifecycle Management isn't new. It applies to any piece of digital content, following it from the beginning, middle and end of creation. With SUSE Manager 4, this idea is applied to software intended for rollout to production systems. Content Lifecycle Management enables you to customize and test packages before updating your production systems. This is especially useful if you need to apply updates during a limited maintenance window.

From within SUSE Manager 4, you can select software channels as sources, adjust them as required for your environment and thoroughly test them before installing them onto your production systems.

With SUSE Manager 4, you can create Content Lifecycles to track such things as monthly patch cycles and kernel versions for Live Patching. To these lifecycles, you can add properties, sources, filters, vendors, base channels and environments. Although you can't directly modify vendor channels, you can clone them and then modify the clones by adding or removing packages and custom patches. You can then assign these cloned channels to test systems to ensure that they work as expected. And, after all tests pass, you can apply the cloned channels to production servers.

Projects

Every Content Lifecycle in SUSE Manager 4 begins its life as a Project (Figure 1).

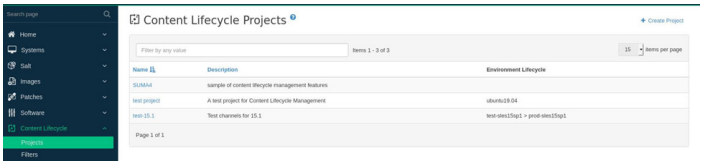

Figure 1. *A list of Content Lifecycle Management projects.*

Working with the lifecycle of a piece of software within a Project makes it easier to control and manage. Within a Project you will create:

- *Project Properties (name, label and description)*
- *Sources (base channel and child channels)*
- *Filters (to define packages and patches)*
- *Environment Lifecycles (to define build environments)*

Project Properties

This section describes the new project. Within the Properties section (Figure 2), you'll want to add information that clearly indicates what the project is. Say, for example, you're testing a particular release of SUSE Linux Enterprise Server. You wouldn't want to give the project the name SUSE Linux Enterprise Server. Why? Which version of SUSE Linux Enterprise Server are you testing in this lifecycle? Instead, be specific. If you're testing version 15 SP1, indicate that.

Figure 2. *Be specific with your Project properties.*

Sources

Sources are the channels to be used for the build of your project. It's important to understand that these sources are pulled from the Channel list that you created under Software in SUSE Manager 4 (Figure 3).

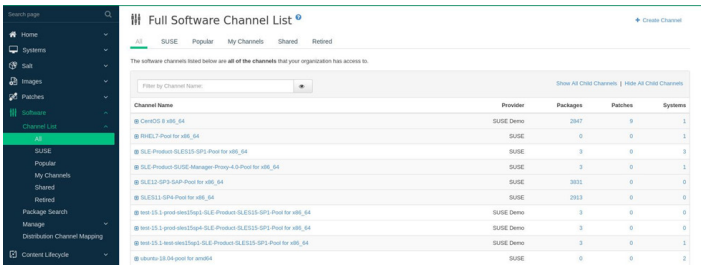

Figure 3. *Sources that can be pulled for Content Lifecycle Management projects.*

Without first creating Software Channels, you won't have any sources to pull for your projects. Once those channels are available, you can then add them with ease (Figure 4).

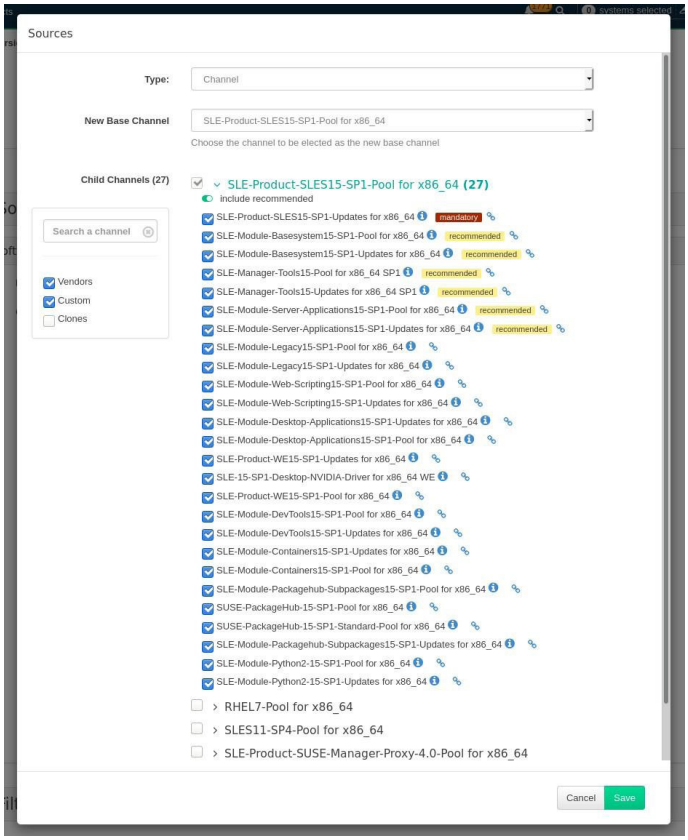

Figure 4. *Available channels to be used as sources for a project.*

You can add as many channels to a source as needed, but you can only have one Base Channel per project. For example, you couldn't add a SUSE Linux Enterprise Server-based source and then add a Ubuntu-based source. However, you can add numerous Child channels to your Base Channel. The Base Channel defines the operating system to be used, such as SUSE Linux Enterprise Server 15 SP1. The Child Channels define various additions to the Base, such as desktop applications, web scripting, various languages, etc.

Filters

Filters enable you to get very specific about what will or will not be built from the sources. For example, you could create filters to deny the addition of specific kernel packages or include specific releases of software, such as a filter to ensure that PHP 7.3 was included in the build (Figure 5).

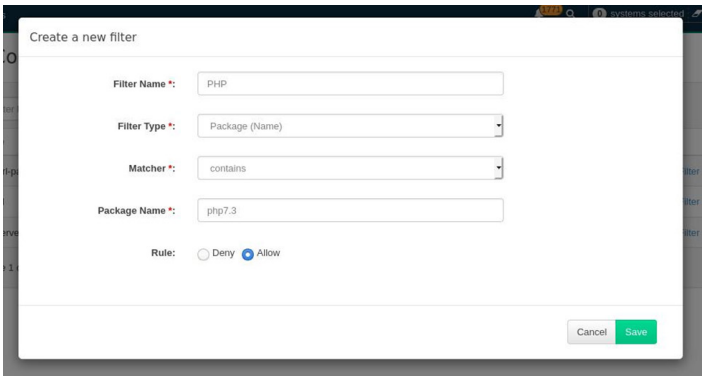

Figure 5. *Creating a filter to include PHP version 7.3 in the build.*

The following is a list of supported filters:

- *Package filtering*
  - *By name*
  - *By epoch*
  - *By version*
  - *By release*
  - *By architecture*
- *Patch filtering*
  - *By advisory name*
  - *By advisory type*
  - *By synopsis*
  - *By keyword*
  - *By date*
  - *By affected package*

Also, multiple matchers are available that can be used with the filter. The filter type determines which matchers are available.

Here is the full list of matchers:

- *Contains*
- *Matches (must take the form of a regular expression)*
- *Equals*
- *Greater*
- *Greater or equal*
- *Lower or equal*
- *Lower*
- *Later or equal*

Environments

A key concept of Concept Lifecycle Management are the Environments. In SUSE Manager 4, Lifecycle Management is achieved through a series of environments that your software channels can move through. Most environment lifecycles include at least test and production environments, but you can have as many environments as you require. For example, you might break down your environments into alpha, beta, testing, pre-production and production.

Or, you could take a completely different to environments. Say, for example, you need to test a patch or a new software rollout on different releases of the same platform, such as SUSE Linux Enterprise Server 15 SP1, SUSE Linux Enterprise Server 15 SP2, SUSE Linux Enterprise Server 15 SP3 and SUSE Linux Enterprise Server 15 SP4, or on different platform altogether, such as SUSE Linux Enterprise Server 15, RHEL8.1, Ubuntu 19.04. With SUSE Manager 4, you can create multiple Environment Lifecycles by clicking the Add Environment button (Figure 6).

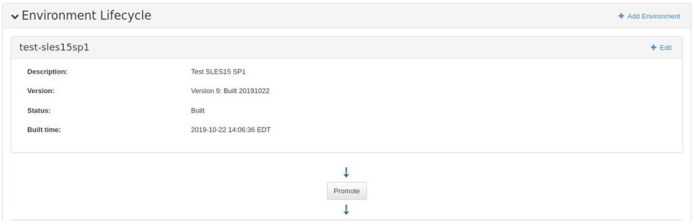

Figure 6. *The Environment Lifecycle section of a project.*

Building

Once you have pieced together your project (defined environments and attached sources and filters), click Build and allow SUSE Manager 4 to do its thing (Figure 7).

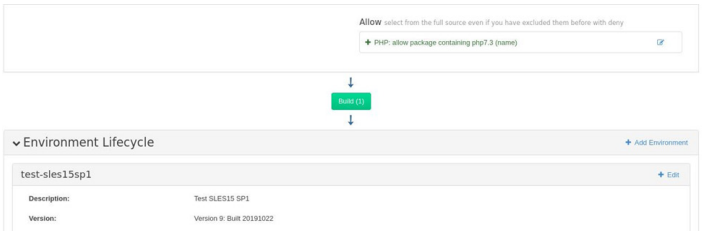

Figure 7. *Building the new project.*

Building applies all filters to the attached sources and clones them to the first environment in the project.

After clicking Build, you are then required to add a Version Message before the build will proceed. This is an important step with Content Lifecycle, as it indicates what is specific to this build (Figure 8). If you don't add specific information, you'll have no idea what this particular build means to the scope of the project.

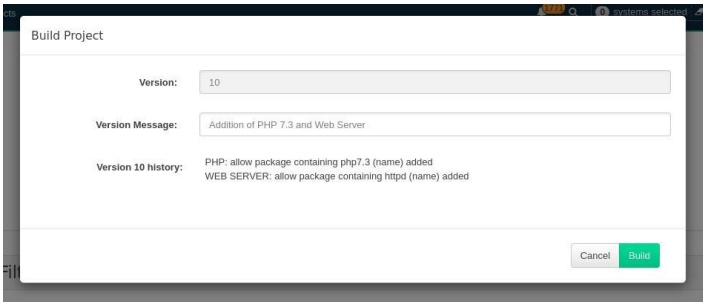

Figure 8. *Information specific to this build.*

Once you've added information specific to the build, click Build. In the Status of the Environment Lifecycles section, you'll see that the Channels are being cloned. Depending on the amount of Child Channels you've included (as well as the number of Filters you've added), this can take some time. After the build is finished, the environment version is increased by one and the built sources (such as software channels) can be assigned to your systems.

Promoting Environments and Assigning Systems

After the project build completes, the built sources can be sequentially prompted to the environments that you created. In the Environment Lifecycle section, locate the environment to promote to its successor and then click Promote (Figure 9).

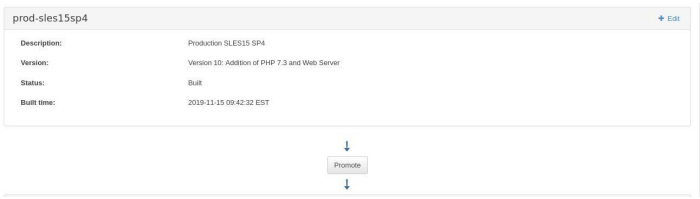

Figure 9. *Promoting an environment.*

When you build and promote content lifecycle projects, it creates a tree of software channels. To add systems to the environment, assign the base and child software channels to your system using the Software Channels tab in the System Details page (from within Systems, Figure 10).

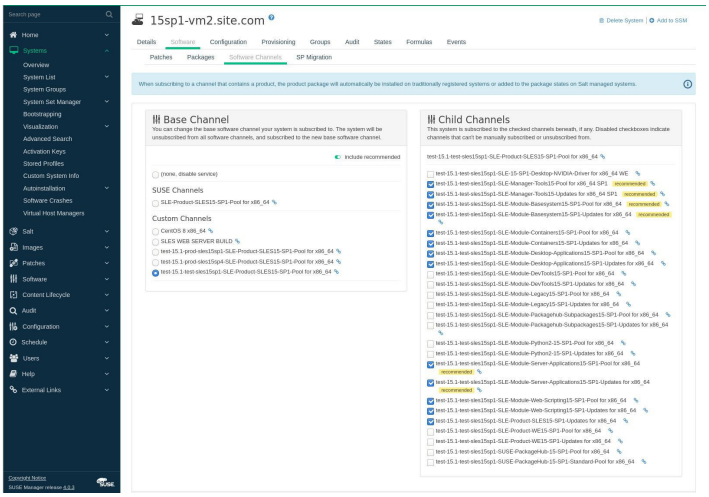

Figure 10. *Adding systems to an environment from within the Systems page.*

NOTE: Newly added cloned channels aren't assigned to systems automatically. If you add or promote sources, you'll need to manually check and update your channel assignments. Automatic assignment is planned to be added to SUSE Manager in a near future version.

Examples of Content Lifecycle Management

At this point, you might be wondering what you can use Content Lifecycle Management for. Here are a few examples.

Monthly patch cycles, which could include:

- *Creating a filter for a project*
- *Adding a filter to a project*
- *Applying the filter to a new build environment*
- *Excluding a patch from a project*
- *Including a patch in a project*

Enhancing a project with live patching. For this, you should always consider these guidelines:

- *Use only one kernel version on your systems. The live patching packages are installed with a specific kernel.*
- *Live patching updates are shipped as one patch.*
- *Each kernel patch that begins a new series of live patching kernels will display the required reboot flag. These kernel patches come with live patching tools. When you have installed them, you will need to reboot the system at least once before the next year.*
- *Only install live patch updates that match the installed kernel version.*
- *Live patches are provided as standalone patches. You must exclude all regular kernel patches with a higher kernel version than the currently installed one.*

Updating a project for next month's patches. For such a Lifecycle Management Project, it is important that you don't change the "Exclude kernel greater than 4.12.14-150.17.1: ..." Filter. This keeps normal kernel-updates from being applied and takes the latest live patches up to the selected month.

Switching to a new kernel version for Live Patching. Prior to SUSE Manager 4, live patching implementation was incredibly ugly. If you tried use zypper up, you could wind up with updates from multiple channels, which can cause problems. With Content Lifecycle Management, you can create a project that includes live patching. Through this process, you can validate that all of your patches work before they are promoted (Figure 11), which enables you to move content independent of system registrations.

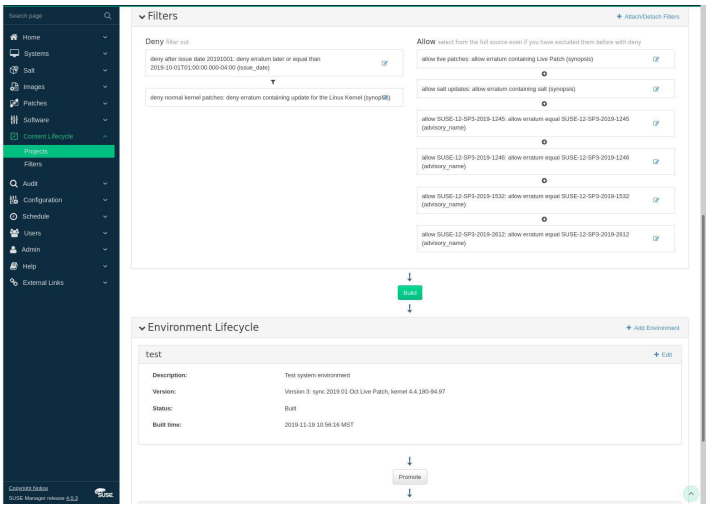

Figure 11. *Using filters to validate patches for live patching a kernel.*

To make this happen, filters are used. For example, you could deny any patch that doesn't meet a specific timeframe or filter out any patch that would require a reboot. This helps to ensure that a build won't fail because of a problematic patch.

In the case of live patching, you can define the time period for a specific kernel lifecycle with live patching and control when you want to reboot. With the zypper lifecycle | grep kgr command, you can search for the specific lifecycle for a kernel live patching time period.

Additional contact information and office locations:
www.suse.com

www.suse.com