

EC2 Gold AMI Automation

CHC Gold AMIs

At CHC, cloud security starts with the AMIs that are modified into the "Gold AMIs", which are then used as the base for applications and services developed at CHC.

The Gold AMIs are configured with:

- A hardened OS
- the AWS CLI installed
- Trend Micro DSA installed
- aws-cfn-bootstrap (to provide cfn-init)
- Filebeat client installed
- Monitoring package installed
- CloudHealth agent installed
- SSH keys are removed by default after AMI creation, but before AMI release

The fully-automated Gold AMI Automation process is diagrammed in Figure 1 below:

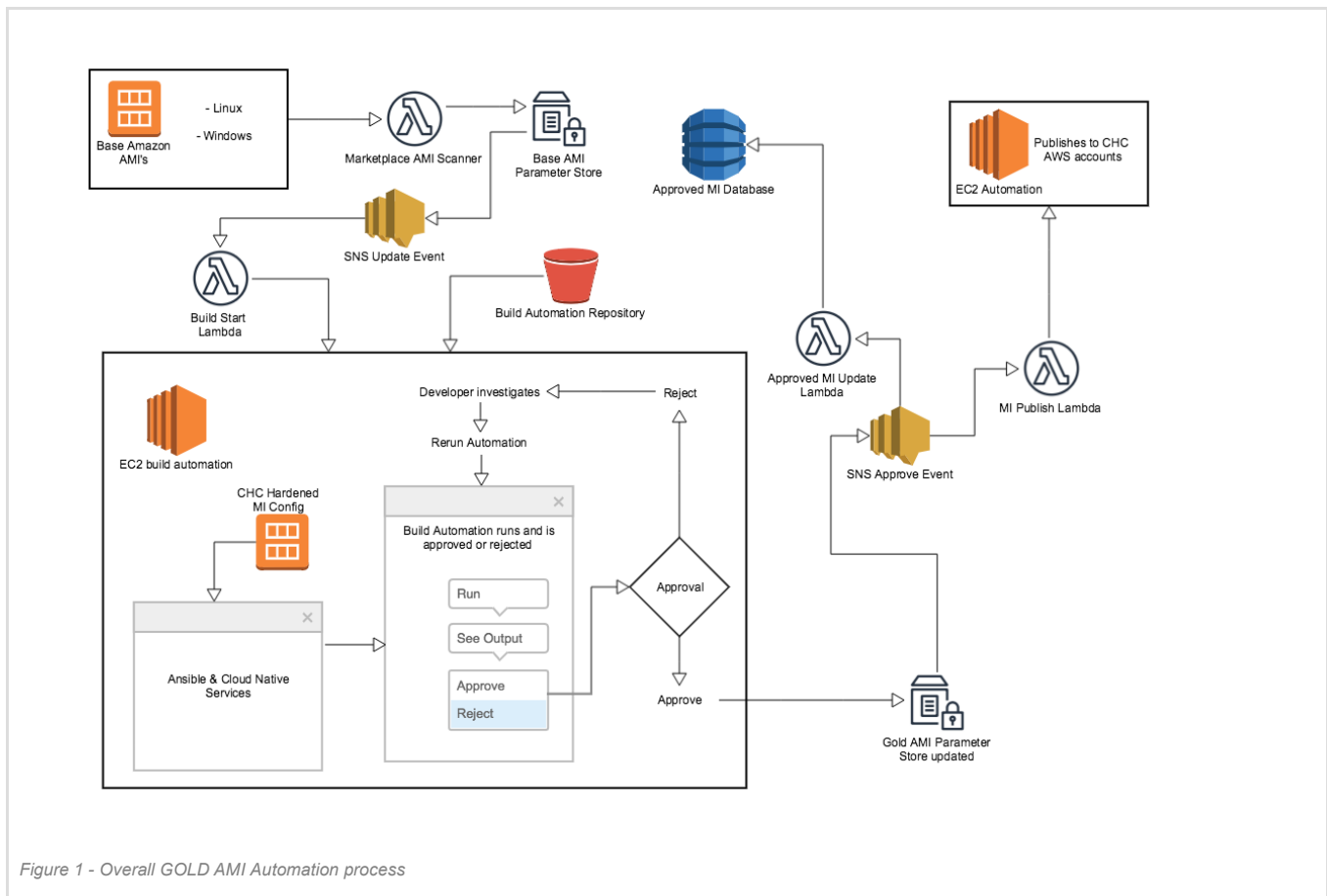


Figure 1 - Overall GOLD AMI Automation process

Building Gold AMIs with Automation

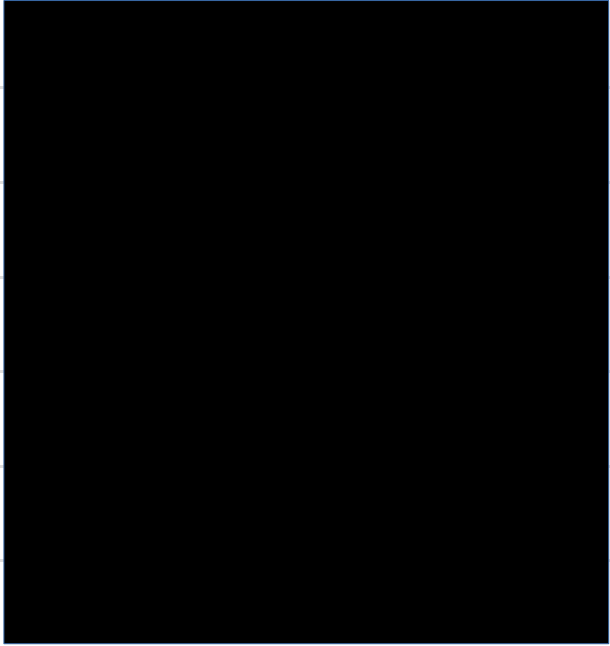
To create a process that is repeatable while also minimizing the opportunity for software flaws or vulnerabilities to be introduced, automation is key. The automation for building CHC Gold AMIs was built in conjunction with AWS resources, and is now in use to keep all images - Linux and Windows - used by CHC in the cloud in compliance with the security policies defined by InfoSec. Starting with the latest AWS Marketplace AMIs for each of our supported OS types, the automation adds the agents and packages required, and tests to make sure that any potential security concerns are addressed (i.e. SSH keys removed, world-writable files are not allowed, etc.).

Initially, the automation was triggered manually by loading a zip file of the [ansible](#) playbooks to create the CHC Gold version of the following supported OS types:

- Amazon Linux
- CentOS 7
- RHEL 7
- Amazon Linux optimized for ECS
- Windows 2012r2
- Windows 2016
- Deep Learning **

** Deep Learning is a special application-specific OS type that includes specific exceptions granted by InfoSec to some of the standard security restrictions. The Service Request ID of the exception is included in the playbooks for this application.

Projects containing the playbooks for each of these OS types are:

AMI Name	Gitlab Project Location
Amazon Linux	
Amazon Linux (ECS optimized)	
RHEL-7	
CentOS-7	
Deep Learning	
Windows 2012r2	
Windows 2016	

The EC2 Gold AMI Automation (EC2G) service *ami-check* monitors the AWS Marketplace looking for new AMI releases. When a new AMI becomes available the EC2G system starts the automation process to create a new base AMI for use within CHC. The automation process can also be started manually.

NOTE: Currently, CIE Ops will manually run the automation process at the end of each month until all of the underlying automation is completed. Additionally, in some cases the automation for Windows images must be run repeatedly to ensure a successful process.

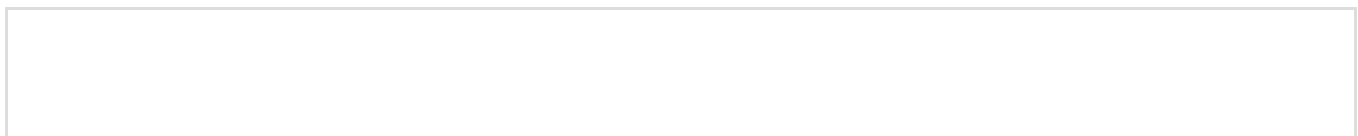
Diagram of the EC2 Gold AMI Automation

Running manually, the first step a developer does is to create a zip file for the OS type, using one of the following names:

- amazon.zip
- amazon-ecs.zip
- centos-7.zip
- rhel-7.zip
- deep-learning-amz.zip
- win2012r2.zip
- win2016.zip

Each of the OS projects contains a script named `launch_automation.sh`, which creates the zip file, uploads it to S3, and then executes the command to run the automation with the required parameters for that specific OS type.

An illustration of this automation is shown below in Figure 2:



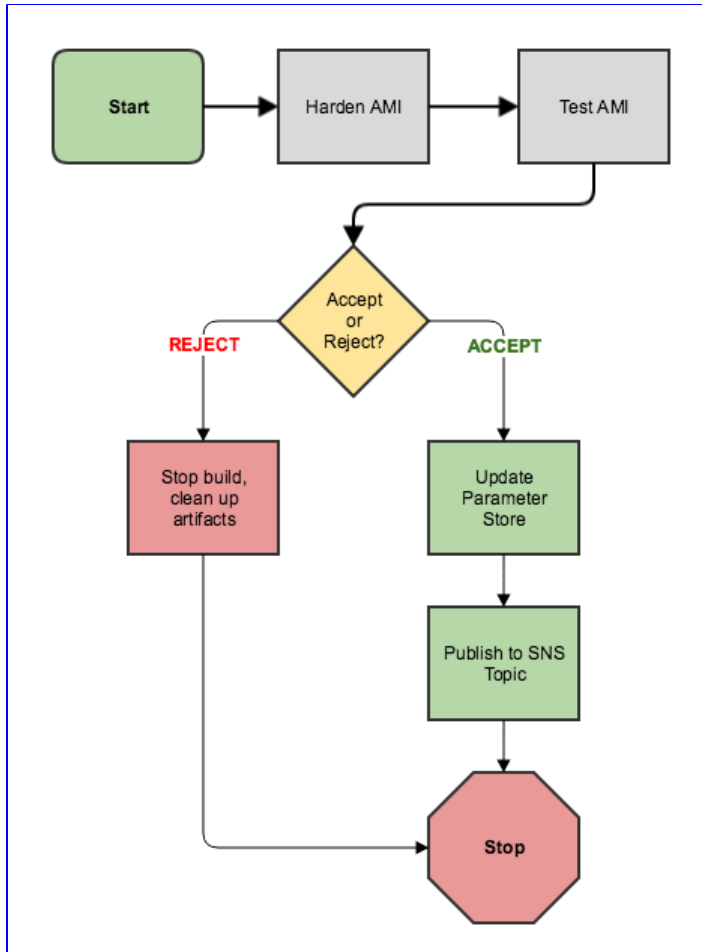


Figure 2 - AMI Check process

If all steps complete successfully, including a compliance check against an instance created with the newly-built AMI, and the build is approved, the new AMI ID is written to the [Parameter Store](#). Checking that all steps have passed and even approving the AMI build may be done through a command-line/script. If the build is not approved - either manually or by script - the Parameter Store value is left unchanged.

Parameter Store

All Base AMI parameter store entries use the following format convention:

```
/nimbus/ami/base/<unique operating system type name>
```

For example:

```
/nimbus/ami/base/centos-7
/nimbus/ami/base/deep-learning-amz
```

When Base parameter store values are changed, they automatically published to the `cie-ami-base` SNS topic.

Gold AMI parameter store entries use the following format convention:

```
/nimbus/gold/<operating system type - linux or windows>/<unique operating system type name>
```

For example:

```
/nimbus/gold/linux/centos-7
/nimbus/gold/windows/win2012r2
```

When Gold parameter store values are changed, they automatically published to the `cie-ami-gold` SNS topic.

AMI-Check

The ami-check service is scheduled to run daily at 4:00 AM PST. Currently, ami-check uses a static list of AMIs to check, but it will be refactored to leverage nimbus-cli and pull the list of AMIs from that. If a new version of the AMI is discovered by ami-check, the appropriate base AMI parameter store value will be updated.

The ami-check project is located in the gitlab repository at: [REDACTED]

Nimbus-AmiAwsLambda

Nimbus-AmiAwsLambda is a target (subscriber) of the `cie-ami-base` SNS topic. It receives the SNS message, parses it for the appropriate values so that the Gold AMI Automation may be triggered for the specific AMI that was updated.

The Nimbus-AmiAwsLambda project is located in the gitlab repository at: [REDACTED]

Build Automation

1. Nimbus-AmiAwsLambda starts the automation, which takes the base AMI and hardens it to CHC security standards.
2. The resulting proposed AMI is then tested to ensure that it complies with CHC security standards.
3. The AMI then waits for manual approval or rejection.
4. If the AMI is approved, the appropriate gold AMI parameter store is updated. Updating the gold AMI parameter store will publish to the `cie-ami-gold` SNS topic.
5. If the AMI is rejected (not approved), the build automation process stops the build, cleans up any artifacts created during the build process, and then terminates. No values in the parameter store are updated.

The Nimbus-AmiAwsLambda project is located in the gitlab repository at: [REDACTED]

Developer Notes

For everything to work correctly, the filenames must match exactly. **DO NOT deviate from the standard names.**

Creation of the base AMI parameter store entries is initially a manual process, as is adding the base AMI parameter store information to ami-check. (Note: this may be refactored to use the nimbus-cli in the future.)

Base AMI parameter wiring is accomplished within the [ami-check](#) project, including:

- Creating the `cie-ami-base` SNS topic
- Creating and putting the CloudWatch rule: indicating updates to the base AMI parameter store value should publish to an SNS topic
- Creating and putting the CloudWatch target: linking the CloudWatch rule and the SNS topic to be published to
- Updating the base AMI parameter store value

Please consult the current code and documentation of [ami-check](#) for more details.

Gold AMI parameter wiring is for the most part accomplished in the `ami-aws-service` project in `gold-automation-document.json`

The following sections of the `gold-automation-document.json` should be noted:

- `AutomationUpdateSsmParamLambdaPolicy`: defines permissions for updating parameter store and creating event rules.
- `AutomationUpdateSsmParamLambdaFunction`:
 - Creates and puts the CloudWatch rule: indicating updates to the Gold AMI parameter store value should publish to an SNS topic
 - Creates and puts the CloudWatch target: links the CloudWatch rule with the SNS topic to publish to
 - Updates the Gold AMI parameter store value

Additional References

[Glossary and Acronym List \(GAL\)](#)
