Salt - Conductor Action types

Overview

In the Salt Conductor framework the central component is the orchestration, aka Conductor. It is a Salt runner, therefore must be executed by the salt-run engine subsystem on the salt master. The conductor cannot and does not run any actions on the local client minions. All that is done within the SaltStack modules. The conductor is simply an extension to salt-run and provides a framework and model to implement Salt as a scalable solution for the enterprise.

The conductor is used to provision, bootstrap, configure and manage minions in the saltstack ecosystem. It is the central automation (that will be exposed through an undetermined interface) in the delivery automation framework and provides end to end delivery of software products, systems and services.

There are 4 high level actions that can be specified when executing conductor. This document will explain these in detail and provide use case examples.

Understanding these action types does require some understanding of the Configuration Model used by Conductor. Refer to provisioning types.

Action Types

The four high level action of conductor are all separate. I.E. you cannot run two different actions within one execution of Conductor. Actions are implemented in the submodules of Conductor. Currently there are two sub modules within the Conductor, Group and Cloud. The four actions described in this document apply to the Group module as this is the primary provisioning submodule or the Conductor salt runner. Thus all command example and such will be using the Group submodule.

The Cloud module supports create and destroy and is specific to use cases of cloud core infrastructure such as vpn's, subnets, etc.... and may or may not be used in our Eliza Enterprise CM system, it's TBD.

The examples shown throughout this document are for knowledge and context only. The roles and product groups used may of may not be actual roles and product groups without the model used in real world.

In practice these commands would be exposed to the user via an interface such as Jenkins, Rundeck ,Salt UI or other. Ultimately the user would execute a task via the interface with specific authorization and options filtered based on our policies. But in any case, the job the user invokes would trigger automation to run these commands on the Salt master.

Create

The Create action in Conductor is supported in the group module as well as the cloud submodule. The Group module implementation is the subject of this document.

Create action is used when you want to create instances in the cloud provider environment (AWS in our case) that are bootstrapped with salt minion software, registered to the master/s, and preconfigured with configuration and software based on a role or a collection of roles. Therefore the create action will require additional parameters. The role or roles applied to instances (role in terms of salt configuration model, not aws roles) will tell the automation what salt states to run to enable an instance, or set of instances in the case of a cluster role, to do something in the environment.

Create action is generic and totally based on the arguments you provide. The next section describes multiple use cases being implemented as conductor commands using the create action.

> The group in 'conduct.group' tells conduction what module to load, the 'group=xxxx' parameter represents the product group. Product groups can be mapped 1:1 to teams or suites/collections of products. Refer to the terminology page. Also note 'salty' is a product group used in these examples.

The format of the Conductor Create command is:

salt-run conduct.SUBMODULE ACTION ARGUMENTS_LIST

Required Arguments:

- group - product group. Can be a team or a collective suite of products owned and managed by one or more teams
- role - similar to a profile/server type. Represents a functional configuration/service/application in the environment.
- pillarenv - salt environment. Maps to pillar branches which map to business environments.
- region - cloud (aws) provider region.

Other arguments may be needed

Examples

Create a single instance of a single role

```
salt-run conduct.group create group=teamA role=activemq count=1
pillarenv=test region=us-east-1a
```

> The count parameter is required as well as pillarenv and region.

Create 4 instances each with the same role

```
salt-run conduct.group create group=teamA role=activemq count=4
pillarenv=test region=us-east-1a
```

Create a single instance that is configured with multiple roles (aka composite role, assume the composite role is configured with 3 different roles in pillar configuration)

```
salt-run conduct.group create group=salty role=test-composite count=1
pillarenv=test region=us-east-1a
```

> The composite role is a role that has multiple base role states as part of its configuration. See pillar example here. In this case and the previous example the same activemq salt state will be used. This demonstrates stacking of roles into composite-roles.

Create instances for a cluster type role 4 members (assume the default in pillar configuration was a cluster with 3 members)

```
salt-run conduct.group create group=salty role=nifi members=4
pillarenv=test region=us-east-1a
```

> Cluster roles are special roles in that they can span multiple instances. Other roles are mapped to single instances with composite roles being single instances with multiple base state roles. For example if you execute conductor create and specify a cluster type in the the role=xxxx parameter, it can create one or more instances depending on the members argument. The entire set of machines will be the 'role'. A role that is a cluster will have special configuration in Pillar to mark it as a cluster. The Conductor use this Pillar configuration when creating the cloud configurations files as well as some other built-in functionality. See pillar example for a cluster configuration here.

Create instances for a cluster type role using the default member quantity in pillar

```
salt-run conduct.group create group=salty role=cassandra pillarenv=test
region=us-east-1a
```

Create a system (a collection of roles and cluster roles)

```
salt-run conduct.group create group=teamX system=test_system
pillarenv=test region=us-east-1a
```

> A 'system' is a provisioning type that can be specified when executing the create action for the group submodule of conductor. System is a powerful provisioning type, but a simple implementation. It basically wraps what would be separate provisioning actions executed in Conductor into one execution. The idea is that a system is a logical set of instances of a variety of roles concluding cluster roles.
>
> An example system could be 2 web server instances, 3 services instances, 3 node clustered database and a 3 node message queue cluster. Without 'system' provisioning type, this would be 4 separate executions of Conductor. The system configuration is designed to create a logical system like the one just described that can be created with a single execution of Conductor.
>
> The system components can be configured to have different quantities and different components within a 'system. But all product specific configuration for the components of the system will be configured outside the 'system' pilllar config. Think of a system configuration in pillar as a simple configuration package of a set of products.
>
> See pillar example of systems here.

Destroy

The Destroy action in Conductor is supported in the group module as well as the cloud submodule.

There is an entire wiki document describing the Destroy action options and use case.

Please refer to: Salt - Targeted teardown of instances, clusters or systems

Examples

refer to: Salt - Targeted teardown of instances, clusters or systems

Upsize

The Upsize action in Conductor is supported in the group module only. It is specific to cluster roles. This it used to 'expand' the members of a cluster. In other words, to add a node member to an existing cluster. Multiple nodes can be added with one execution of upsize action.

Required Arguments:

- group - product group. Can be a team or a collective suite of products owned and managed by one or more teams
- role - similar to a profile/server type. Represents a functional configuration/service/application in the environment.
- ***clusterid*** - this must be the valid cluster id for the cluster to be upsized. When clusters are created with Conductor, specific salt grains are created and added to each cluster member in the minion metadata. Some of these grains are added as part of the salt-cloud bootstrap vis the cloud configuration files Conductor auto-generates based on pillar config. Some grains are added by Conductor code once the new cluster has become bootstrapped. The clusterid required for the upsize action is the ***productgroup.product.cluster.id*** salt grain. *Be sure to use this one when executing upsize action*.
- pillarenv - salt environment. Maps to pillar branches which map to business environments.
- region - cloud (aws) provider region.

Optional Arguments:

- **members** is an optional argument. This is how you specify how many new node members to add to the existing cluster. If NOT specified, one new node member will be created and added to the cluster. This defaults to 1.

Examples

Upsize a cassandra cluster with 2 new node members

```
salt-run conduct.group upsize group=salty role=cassandra clusterid=1
members=2 pillarenv=test region=us-east-1a
```

Upsize a nifi cluster for polaris product group in test environment by one (uses default member count)

```
salt-run conduct.group upsize group=polaris role=nifi clusterid=1
pillarenv=test region=us-east-1a
```

Demonstrate create and upsize a cluster to show grains being used
Create new cluster:

salt-run conduct.group create group=salty role=nifi count=1 pillarenv=test region=us-east-1a

Test to show new members:

salt '*' grains.item cluster.members

salt '*' grains.item cluster.members.ip

(get new salty.nifi.cluster.id grains value)

Upsize by 1:

salt-run conduct.group upsize group=salty role=nifi clusterid=1 members=1 pillarenv=test region=us-east-1a

Upsize by 2:

salt-run conduct.group upsize group=salty role=nifi clusterid=1 members=2 pillarenv=test region=us-east-1a

Test to show change to members grains:

salt '*' grains.item cluster.members

salt '*' grains.item cluster.members.ip

Should see new members and pre-existing members updated grains

Downsize

The downsize action in Conductor is specific to cluster roles, and it is only supported in the group module as well. This is used to remove a node

from an existing cluster. There are usually one of two scenarios that cause nodes to be removed from clusters. The downsizing of the cluster to avoid unused resources. Or to remove a node instance that is scheduled to be taken down by the cloud provider (aws).

Because there are two different use cases for needing to downsize a cluster, there are a couple different ways to execute this action in Conductor.

Possible Arguments:

- group - product group. Can be a team or a collective suite of products owned and managed by one or more teams **(required)**
- role - similar to a profile/server type. Represents a functional configuration/service/application in the environment. **(required unless node=xxx is specified)**
- ***clusterid*** - this must be the valid cluster id for the cluster to be upsized. When clusters are created with Conductor, specific salt grains are created and added to each cluster member in the minion metadata. Some of these grains are added as part of the salt-cloud bootstrap vis the cloud configuration files Conductor auto-generates based on pillar config. Some grains are added by Conductor code once the new cluster has become bootstrapped. The clusterid required for the upsize action is the ***productgroup.product.cluster.id*** salt grain. *Be sure to use this one when executing upsize action*. **(required when using role=xxx and not node=xxx)**
- ***internalrole*** - each node member in a cluster has an internal role. This will default to secondary if not specified. (cluster internal role is a little complex design in the framework. See the Cluster Internal Role Footnote **(optional and valid only if using role and clusterid instead of node).** This parameter maps to the internal.role grain used when Conductor creates a cluster.
- ***node*** - specify the exact instance name to remove from the cluster. This is used for the use case where we need to remove a particular node member regardless of the internal role. For example, the cloud provider has alerted that this instance is attached to a host that is being deprecated or taken offline for other reasons.
- pillarenv - salt environment. Maps to pillar branches which map to business environments. **(required)**
- region - cloud (aws) provider region. **(required)**

Examples

Downsize a cluster by removing a specific member node

```
salt-run conduct.group downsize group=salty
node=salty-nifi-02.clid-1.us-east-1a.test.foobar.com pillarenv=test
region=us-east-1a
```

Downsize a cluster by one member using default 'secondary' internal role

```
salt-run conduct.group downsize group=salty clusterid=1 role=nifi
pillarenv=test region=us-east-1a
```

> If the above command returns a message saying no valid member found, it means there is no secondary members of the cluster remaining to be removed. In this case, re-run the command and pass the internalrole=xxxx parameter and specify a valid internal.role.

Downsize a cluster by one member specifying the internal role

```
salt-run conduct.group downsize group=salty clusterid=1 role=nifi
internalrole=secondary pillarenv=test region=us-east-1a
```

Cluster Internal Role Notes

Within a Conductor provisioned cluster, there is always a primary/manager and secondary internal.role grain. If the cluster is one member node, it will always be primary. Some cluster technology requires additional roles such as the arbiter role in a mongodb cluster. For the most part most cluster technologies have either no specific roles (i.e all members are equal and perform the same duties), self elect a leader which would be like

no specific role use case. Even when the technology does not require an internal designation of members roles, often during a continuous deployment with provisioning, a task needs to be performed on one cluster node and not the others. These are a few examples of why the Conductor Model for clustered roles uses internal-role designation of primary and secondary even if the technology itself doesn't need it. Even if the deployment/provisioning of the cluster doesn't need to do a special task on one node, it's a non-invasive requirement in the model. If a cluster is created with one member, it will always be the primary. There is always only one primary member in a cluster and all remaining members are salted as secondary internal.role. Unless, as with mongodb, we configure a 3rd internal role in the cluster pillar configuration. Understanding this design, you can see that we could always define any internal.role we want regardless of the technology requiring it without affecting the function of the cluster. It's a management configuration that is designed to make creating salt states for clustered technology products very generic.

This is out of scope for this document, however. In short the salt state development should go hand in hand with the decided Pillar model configuration for all clustered technologies. Please refer to the salt state and pillar design documents.