CHEF COOKBOOK FOR AVI NETWORKS

Chef automation software helps devops to turn their infrastructure into code to manage various stages of the IT infrastructure lifecycle, including the provisioning, patching, configuration, and management of operating system and application components across enterprise data centers and cloud infrastructures.

Avi Networks Cloud Application Delivery Platform (CADP) natively includes an integrated Chef client, which allows configuration synchronization and control from a Chef Server. This solution brief requires a basic familiarity with Chef automation tools and Avi Networks CADP.

Common Use Cases

- Configuration management of virtual servers, servers and pools
- Backup and restore configuration
- Automating config creation through auto-generated config files

Chef Server Configuration

Chef client can interact with the Avi Controller through the CLI interface.

Create Chef recipe:

```
repo_path = "root"
config_script = "/opt/avi/python/bin/utils/config_import.py"
new_config = "vs.json"

remote_file "Copy config file" do
  path "/tmp/#{new_config}"
  source "file:///#{repo_path}/#{new_config}"
  owner 'root'
  group 'root'
  mode 0755
end

execute 'execute_file' do
 cwd '/tmp'
 command "python #{config_script} --command 'import configuration file /tmp/#{new_
config}'"
end
```

NOTE: Please change repo_path and new_config to appropriate path and config file name in your environment.

Node Definitions

Add the Avi Controller to the node definitions of the Chef:

```
{
  "run_list": [ "recipe[avi]" ]
}
```

Example 1: Create a Virtual Service

To create a virtual service within Avi, you will need a snippet of the JSON code for a sample virtual service. To grab the code from an existing virtual service within your setup, run the following command from the CLI shell of your Avi Controller:

```
export virtualserver <VS_NAME>
```

The following is an example of a JSON snippet of a VS configuration. Modify the names and addresses prior to using Chef to create a new virtual service on your Controller.

```
{
    "VirtualService": [
        {
            "description": "",
            "name": "My Application",
            "address": "10.10.1.10",
            "ip_address": {
                "addr": "10.10.1.10",
                "type": "V4"
            },
            "enabled": true,
            "services": [
                {
                    "port": 80,
                    "enable_ssl": false
                }
            ],
            "application_profile_ref": "admin:System-HTTP",
            "network_profile_ref": "admin:System-TCP-Proxy",
            "pool_ref": "admin:NginxPool",
            "ssl_key_and_certificate_refs": [],
            "analytics_policy": {
                "full_client_logs": {
                    "enabled": true,
                    "duration": 30
                },
                "client_log_filters": [],
                "client_insights": "ACTIVE",
                "metrics_realtime_update": {
                    "enabled": true,
                    "duration": 60
                }
            },
            "enable_autogw": false,
            "tenant_ref": "admin",
        }
    ],
    "Pool": [
        {
            "description": "",
            "name": "p1",
            "default_server_port": 80,
            "graceful_disable_timeout": 1,
```

```
                    "connection_ramp_duration": 299,
                    "max_concurrent_connections_per_server": 10000,
                    "health_monitor_refs": [
                        "admin:System-Ping",
                        "admin:System-TCP"
                    ],
                    "servers": [
                        {
                            "ip": {
                                "addr": "10.40.21.62",
                                "type": "V4"
                            },
                            "hostname": "nginx-1.avi.local",
                            "enabled": true,
                            "ratio": 1
                        },
                        {
                            "ip": {
                                "addr": "10.40.21.61",
                                "type": "V4"
                            },
                            "hostname": "nginx-2.avi.local",
                            "enabled": true,
                            "ratio": 12
                        }
                    ],
                    "lb_algorithm": "LB_ALGORITHM_LEAST_CONNECTIONS",
                    "use_service_port": false,
            "inline_health_monitor": false,
            "networks": [],
            "tenant_ref": "admin",
            "certificate": []
        }
    ],
}
```

Add new service port to a Virtual Service

Updates to a Virtual Service can be made by modifying the contents of a JSON config file. For the example of adding another service port (443) to an existing Virtual Service, add following lines at top of services list in lb.json file -

```
                    {
                        "port": 80,
                        "enable_ssl": false
                    },
```

The new object will look like -

```
        "services": [
                    {
                        "port": 80,
                        "enable_ssl": false
```

```
                },
                {
                    "port": 443,
                    "enable_ssl": true
                }
            ],
```

The next run of the chef client on the Avi Controller will add the new service port to Virtual Service. Removing service ports or other configuration can be done with remove the corresponding configuration from JSON file.

Add SSL certificate to a Virtual Service

To add an SSL certificate, add following lines to lb.json file.

```
ssl_profile_ref: "admin:Standard",
ssl_key_and_certificate_refs: ["admin:System-Default-Cert"]
```

This will use System-Default-Cert while serving requests coming on an SSL enabled service port.

Advanced Usage

The Avi module can be customized to run all commands supported by our CLI. This can be done by adding multiple lines:

```
execute 'execute_file' do
 cwd '/tmp'
 command "python #{config_script} --command 'CLI_COMMAND_TO_EXECUTE"
end
```

About Avi Networks

Avi Networks is the Cloud Application Delivery Company. The Avi Networks Cloud Application Delivery Platform (CADP) brings the benefits of hyperscale application delivery to enterprises at any scale. With a unique analytics-driven and distributed application delivery architecture – HYDRA, the Avi Networks solution guarantees end-user application experience for on-premise and cloud-based applications. Please visit us at www.avinetworks.com or follow us on twitter @avinetworks

Avi Networks
945 Stewart Drive · Suite 150
Sunnyvale, California 94085
408.628.1300
www.avinetworks.com