Steps to Migrate a BaaS Role

1. git clone git@github.wellsfargo.com:mrc-eng/mrc_dev.git
2. cd mrc_dev
3. git checkout factory_dev_capgem
4. For the purposes of the rest of this document, this directory (the root of the control repository) will be referred to as <control_repo>.
5. Extract the zip of the original capgemini code base <TODO: Add link>
6. Add one or more hostname entries for the role to the bunsen.yaml file. You can get the information for this from the capgemini-immediate-code/vagrant/config/nodes-<accelerator>.json file, where <accelerator> depends on the role being migrated.

In this example, the role api_manager_database is being migrated, so we look in the nodes-apim.json file for the details as this is part of the apim (api manager accelerator). In this particular case, there are multiple nodes defined in the file with this role, but for initial migration purposes we really just need to add one, so we choose the initial.

```
"apim-api-manager-database-initial": {
        ":base": "immediate/centos-7-0-x86-64",
        ":ip": "192.168.99.115",
        ":role": "api_manager_database",
        ":memory": 1024,
        ":cpus": 1,
        "ports": [],
        "scripts": [
          {
            "inline": "/usr/local/bin/bootstrap-consul.sh $1 $2 $3 $4 $5 $6",
            "args": "true 192.168.99.115 false 192.168.99.10"
          },
          {
            "inline": "sudo systemctl stop firewalld"
          },
          {
            "inline": "sudo systemctl disable firewalld"
          },
          {
            "inline": "sudo sed -i 's/.*wsrep_node_address.*/wsrep_node_address =
apim-api-manager-database-initial.node.core.local/g' /etc/my.cnf.d/server.cnf"
          },
          {
            "inline": "sudo sed -i
's/.*wsrep_node_incoming_address.*/wsrep_node_incoming_address =
apim-api-manager-database-initial.node.core.local/g' /etc/my.cnf.d/server.cnf"
          },
          {
            "inline": "sudo sed -i 's/.*wsrep_node_name.*/wsrep_node_name =
apim-api-manager-database-initial.node.core.local/g' /etc/my.cnf.d/server.cnf"
   },
          {
            "inline": "sudo sed -i
's/.*wsrep_sst_receive_address.*/wsrep_sst_receive_address =
apim-api-manager-database-initial.node.core.local/g' /etc/my.cnf.d/server.cnf"
          },
          {
            "inline": "sudo service mysql stop"
          },
          {
            "inline": "sudo service mysql start --wsrep_cluster_address=gcomm://"
          }
        ],
        ":provisioner": "puppet-masterless"
    },
```

In the bunsen.yaml entry:

- The hostname can generally stay the same.
- The box is the equivalent to the "base" entry, and this will always be box/rhel_72_20152 for right now.
- The IP address can be copied over, changing the third octet from 99 to 51.
- The server_role is the same as the role except that role:: should be prepended to the value from the nodes file since it is fully qualified in Bunsen.
- The other entries can be disregarded for the purposes of the bunsen.yaml file.

Here is an example of the above entry converted to the bunsen.yaml file.

```
- :hostname: apim-api-manager-database-initial
  :box: 'box/rhel_72_20152'
  :guest: :linux
  :server_role: "role::api_manager_database"
  :ip: 192.168.51.115
```

6. Copy the role file from **capgemini-immediate-code/control/modules/role/manifests/<role>.pp to <control_repo>/site/role/manifests/<role>.pp**

7. Modify the role to include required items.

This is the role before modification.

```
class role::api_manager_database (
) {
    class { '::profile::base':
      stage => repos,
    }
    class { '::profile::ap::consul':
      stage => dns,
    }
    class { '::profile::ap::serverspec_test':
      stage => testing,
    }
    include ::profile::sm::sensu
    include ::profile::apim::mariadb_cluster
}
```

This is the role after modification. The following elements were added.

- Definition of the run stages repos, dns, post, and testing.
- Dependency ordering of the run stages
- Declaration of the profile::factory_layered_products profile in the repos stage.
- profile::base adds a require upon Class['profile::factory_layered_products']
- Declaration of the profile:😛ost profile in the post stage. The post profile is intended to perform actions that were previously done in the scripts/inlines defined in the node definitions in the immediate framework. For example, bootstrap_consul.sh is a standard execution in this profile for all roles. There may be script/inlines in your node that need to be evaluated for code changes to include them in the puppet run. There will be more detail on this in a subsequent step.

```
class role::api_manager_database (
) {
  stage { 'repos': }
  stage { 'dns': }
  stage { 'post': }
  stage { 'testing': }
  Stage['repos'] -> Stage['main'] -> Stage['dns'] -> Stage['post'] -> Stage['testing']
  class { 'profile::factory_layered_products':
    stage => repos,
  }
  class { '::profile::base':
    stage => repos,
    require => Class['profile::factory_layered_products'],
  }
  class { '::profile::ap::consul':
    stage => dns,
  }
  class { '::profile::post':
    stage => post,
  }
  class { '::profile::ap::serverspec_test':
    stage => testing,
  }
  include ::profile::sm::sensu

  include ::profile::apim::mariadb_cluster
}
```

8. Take note of the profiles declared by the role. Determine if any of them have not yet been migrated by looking under **<control_repo>/site/profile/manifests** to see if they have already been moved there. If any profiles are missing then copy them from **capgemini-immediate-code/control/modules/profile/manifests** into the equivalent control repository location. In this example, profile::apim::mariadb_cluster had not previously been migrated, so it was copied from **capgemini-immediate-code/control/modules/profile/manifests/apim/mariadb_cluster.pp** to **<control_repo>/site/profile/manifests/apim/mariadb_cluster.pp**

9. Examine any new profiles to determine what dependencies they have, and if any files or templates need to be migrated over specific to the new profiles. Look for puppet fileserver references like puppet:///modules/profile/apim/mariadb/reg_um.sql or template function calls. Typically, these will reference files or templates within the profile module, so they can be copied from **capgemini-immediate-code/control/modules/profile/files** for files and **capgemini-immediate-code/control/modules/profile/templatescapgemini-immediate-code/control/modules/profile/templates** for templates.

For example, this profile had the file references puppet:///modules/profile/apim/mariadb/apimgt.sql
and puppet:///modules/profile/apim/mariadb/reg_um.sql, so these files were copied from **capgemini-immediate-code/control/modules/profile/files/apim/mariadb** to **<control_repo>/site/profile/files/apim/mariadb**. Templates work similarly except they are under templates instead of files.

Another situation that may arise with new profiles is that there may be issues with the new modules in our environment (trying to access external packages or files, etc.) that may need to be modified in order to have a minimum viable product. These modules will be addressed properly in the module work, but for now we make the minimum changes necessary to get them to work.

10. Under **<control_repo>/site/profile/files/common/tests/serverspec/properties/packer** and **<control_repo>/site/profile/files/common/tests/serverspec/properties/terraform,** rename the files for the role in each of these directories. The directories have already been pre-populated although with the old file naming, but if any files are missing check for them in the original capgemini source.

git mv api_manager_database_properties.yml role__api_manager_database_properties.yml

11. Run bunsen in order to pick up the new node that was added to bunsen.yaml

12. Add at least minimal spec tests (using rspec-puppet). Minimal means should compile.with_all_deps and making sure the role class is in the catalog. Copy **<control_repo>/spec/classes/role__service_discovery_server_spec.rb** to the appropriate name for the new role, such as **<control_repo>/spec/classes/role__api_manager_database_spec.rb** and then modify the code to suit testing the new role (change role name, any facts if needed). The spec tests can be executed by using:

```
export BUNSEN_PUPPET_VERSION=2015.2
bunsen master up
bunsen test spec
```

13. **TODO:** More detailed example of an inline script migration to puppet, e.g. sed commands/etc. for this role, others

14. Migrate the hieradata for the role. Copy from **capgemini-immediate-code/control/hieradata/role/<role>.yaml** to **<control_repo>/hieradata/role__<role>.yaml**.

For example: **cp capgemini-immediate-code/control/hieradata/role/api_manager_database.yaml <control_repo>/hieradata/role__api_manager_database.yaml**

If common::users and/or common::groups are present in the hieradata, then convert them to the format expected by the new version of common, which integrates with boks::user_group.

For the example, here is the original hieradata for common::users and common::groups.

```
common::users:
  mysql:
    ensure:  present
    comment: 'MySQL server'
    gid:     mysql
common::groups:
  mysql:
    ensure: present
```

The common::groups should be removed and the entries should match the following format (add basconsl to all roles). The user and group names must be converted to match the names of the service accounts that were created in BoKS. Many attributes of users and groups can not be managed directly by puppet under BoKS, and in fact under BoKS this code will simply verify that BoKS has pushed the user and group to the box, so the other attributes are generally discarded.

```
common::users:
  basmysql:
    groups: basmysql
  basconsl:
    groups: basconsl
```

15. Determine if there is any module-level hieradata needed by this profile that hasn't been migrated by examining the modules used by the profiles, and see if there are any **capgemini-immediate-code/control/hieradata/module/<module>.yaml files** that haven't been migrated to the **data** directory of the module as common.yaml with a standard hiera.yml.

https://github.wellsfargo.com/tracers/consul/tree/master/data can provide an example of what this migration looks like on a module where it has already been done.

16. Test the node(s) for the new role with vagrant.

vagrant up apim-api-manager-database-initial

All of the serverspec tests should pass in the puppet run output.   If you brought up the consul server ahead of this (**vagrant up ap-service-discovery-server-0)**, then you should be able to check the Consul UI and see the registration of the new node and corresponding service(s) in the UI.

17. Ensure that the role is idempotent with second provision by running **vagrant provision apim-api-manager-database-initial** There should be no changes applied by the catalog other than the execution of the serverspec tests, which should report success.

18. **TODO: Document testing the role in PCE (initially nova_tools, later Terraform)**