

Advanced Hiera Activity

1. Review the previous DPK Guided Lab setup
 2. Make changes to DPK to enable `hiera_hash`
 3. Change Process Scheduler feature settings
 4. Test and Apply changes
-

Review Custom DPK

Review the customized DPK setup that was completed in the DPK Guided Lab

1. Open the folder `c:\psft\dpk\puppet` in VisualStudio Code.
2. Open the console by pressing `Ctrl+``.
3. Checkout the `Advanced DPK` starting code.

A screenshot of a terminal window with a dark blue background and light blue text. The terminal shows two lines of commands: `PS> git fetch` and `PS> git checkout adv/master`. The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

```
PS> git fetch
PS> git checkout adv/master
```

4. Find the hiera configuration file `hiera.yaml` and review.
5. Find the yaml files under `production\data` and review.
6. Find the custom `site.pp` file at `production\manifests` and review.
7. Find the `dpk_lab` module under `production\modules` and review.

Make Changes for `hiera_hash`

1. In VSCode, right click on the `production/modules/pt_profile` folder and `Find in Folder`.

- Find all `hier('prcs_domain_list')` and replace with `hier_hash('prcs_domain_list')`.

```
Search: hier('prcs_domain_list')
Replace: hier_hash('prcs_domain_list')
Enter to search
Click 'Replace All'
```

- Open the `hier.yml` file.
- Add a new line to the bottom of the file to update merge behavior.

```
:merge_behavior: deeper
```

Change Process Scheduler feature settings

- Open the `production/data/psft_configuration.yml` file.
- Find the `prcs_domain_list:` hash and copy the whole section.

```
prcs_domain_list:
  "%{hier('prcs_domain_name')}":
    os_user: "%{hier('domain_user')}"
    ps_cfg_home_dir: "%{hier('ps_config_home')}"

  db_settings:
    db_name: "%{hier('db_name')}"
    db_type: "%{hier('db_platform')}"
    db_opr_id: "%{hier('db_user')}"
    db_opr_pwd: "%{hier('db_user_pwd')}"
    db_connect_id: "%{hier('db_connect_id')}"
    db_connect_pwd: "%{hier('db_connect_pwd')}"

  config_settings:
    Process Scheduler/PrsServerName: "%{hier('prcs_domain_id')}"
    Security/DomainConnectionPwd: "%{hier('domain_conn_pwd')}"

  feature_settings:
    MSTRSRV: "Yes"
    APPENG: "Yes"
```

- Open the `production/data/psft_customizations.yml` file.
- Go to the bottom of the yml file and paste the `prcs_domain_list` hash.
- Delete everything under `"%{hier('prcs_domain_name')}":` and before `feature_settings:`.
- Change the `APPENG` value to `No`.

```
prcs_domain_list:
```

```
"%{hiera('prcs_domain_name')}":  
  feature_settings:  
    MSTRSRV:    "Yes"  
    APPENG:     "No"
```

Test and Apply Changes

1. Open console again, using Ctrl+ `.
2. Test the results of using the old hiera method.

```
PS> hiera prcs_domain_list --config=./hiera.yaml  
{ "psftdb" => { "feature_settings" => { "MSTRSRV" => "No", "APPENG" => "Yes" } } }
```

3. Test the results of using the new hiera_hash method.

```
PS> hiera --hash prcs_domain_list --config=./hiera.yaml  
{ "psftdb" =>  
  { "os_user" => "psadm2",  
    "ps_cfg_home_dir" => "c:/psft/cfg",  
    "db_settings" =>  
      { "db_name" => "PSFTDB",  
        "db_type" => "ORACLE",  
        "db_opr_id" => "VP1",  
        "db_opr_pwd" => "VP1",  
        "db_connect_id" => "people",  
        "db_connect_pwd" => "people" },  
    "config_settings" =>  
      { "Process Scheduler/PrCSServerName" => "PRCS",  
        "Security/DomainConnectionPwd" => "Passw0rd_" },  
    "feature_settings" => { "MSTRSRV" => "Yes", "APPENG" => "No" } } }
```


4. Check to see if the Process Scheduler and `PSAESRV` are running.

```
PS> psa summary  
PS> Get-Process -Name PSAESRV
```

5. Run `puppet apply` to apply the changes.

```
PS> cd c:\psft\dpk\puppet  
PS> puppet apply -e "include ::pt_profile::pt_prcs" --confdir=.
```

6. Start the Process Scheduler and check again to see if `PSAESRV` is running.



PS> psa start prcs
PS> Get-Process -Name PSAESRV

Puppet Environments Activity

1. Create environments and common structure
2. Update config files
3. Create environments
4. Apply changes to new environment

Create environments and common structure

1. Open the folder `c:\psft\dpk\puppet` in VisualStudio Code.
2. Drag all the folders from under `production` up a level to under `puppet`.
3. Remove the `production` folder.
4. Create a new `environments` folder under the `puppet` folder.
5. Create a new `environments\production` folder.
6. Create a new `environments\production\manifests` folder.
7. Create a new blank `site.pp` file under `environments\production\manifests`.
8. Go to the `puppet\data` folder.
9. Create a new `environment` folder under the `data` folder.
10. Validate new structure.

```
PS> Get-childitem | Select Name
```

```
Name
----
data
environments
manifests
modules
secure
ssl
.gitignore
environment.conf
hiera.yaml
```

Update config files

1. Open the `hiera.yaml` file.
2. Update the `:hierarchy:` hash to include our new `environment` data.

```
:hierarchy:  
- environment/%{:environment}  
- defaults  
- psft_customizations  
- "%{:app}"  
- pwd  
- common  
- psft_unix_system  
- psft_deployment  
- psft_configuration  
- psft_patches
```

3. Update the `:datadir:` value for both `:yaml:` and `:eyaml:`, removing `production` from path.

```
:yaml:  
  :datadir: c:\psft\dpk\puppet\data  
  
:eyaml:  
  :datadir: c:\psft\dpk\puppet\data
```

4. Open the `puppet.conf` file.
5. Update the `environmentpath` setting and add a `basemodulepath` setting.

```
environmentpath=c:\psft\dpk\puppet\environments  
hiera_config=c:\psft\dpk\puppet\hiera.yaml  
basemodulepath=c:\psft\dpk\puppet\modules
```

Create Environments

1. Create paths for new environments `env0` and `env1`.

```
PS> mkdir -p environments/env0/manifests  
PS> mkdir -p environments/env1/manifests
```

- Copy `manifests/site.pp` file to new environments `manifests` folders.

```
cp manifests/site.pp environments/env0/manifests/site.pp
cp manifests/site.pp environments/env1/manifests/site.pp
```

- Edit the `environments/env1/manifests/site.pp` file to use a Process Scheduler only role.

```
node default {
  include ::pt_role::pt_app_prcs
}
```

- Under the `data/environment` folder, create an `env0.yaml` file.

- Update the `env0.yaml` file to set the default `ps_config_home` variable.

```
---
ps_config_home:      'c:/psft/cfg'
```

- Under the `data/environment` folder, create an `env1.yaml` file.

- Update the `env1.yaml` file to set a new `ps_config_home` and `env_type`.

```
---
ps_config_home:      'c:/psft/cfg-env1'
env_type:             'midtier'
```

- Apply our changes by running `puppet apply` for the `env1` environment only.

```
PS> cd c:\psft\dpk\puppet
PS> puppet apply environments\env1\manifests\site.pp --confdir=. --environment=env1
```


- While `puppet apply` is running, open FireFox and login to PeopleSoft to confirm `env0` is not effected.

```
- http://localhost:8000/psf/ps/EMPLOYEE/?cmd=login
- VP1/VP1 or PS/PS
```

- Validate that a new Process Scheduler was created for `env1`.

```
PS> $env:PS_CFG_HOME='c:/psft/cfg-env1'
PS> psa summary
```

- Validate that `env0` is still in place and running.



```
PS> $env:PS_CFG_HOME='c:/psft/cfg'  
PS> psa summary
```

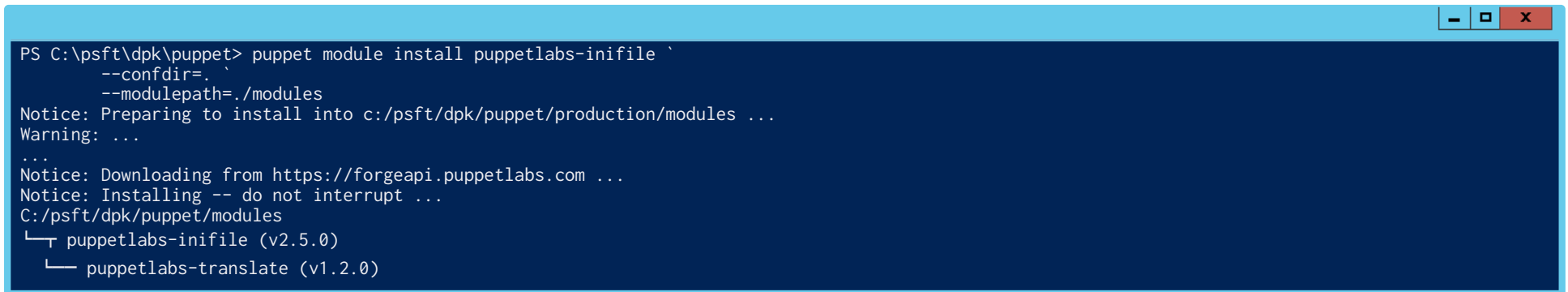

Puppet Modules Activity

1. Install and use the inifile module
 2. Install and use an io module
 3. Make an update to an io module
 4. Create a Pull Request for an io module
-

The inifile Module

We will install a Puppet module: `puppetlabs-inifile`. The `inifile` module makes it easy to update a standard configuration file.

1. Install the module.



```
PS C:\psft\dpk\puppet> puppet module install puppetlabs-inifile `
  --confdir=. `
  --modulepath=./modules
Notice: Preparing to install into c:/psft/dpk/puppet/production/modules ...
Warning: ...
...
Notice: Downloading from https://forgeapi.puppetlabs.com ...
Notice: Installing -- do not interrupt ...
C:/psft/dpk/puppet/modules
└─ puppetlabs-inifile (v2.5.0)
  └─ puppetlabs-translate (v1.2.0)
```

If you run into an HTTPS error, download the RubyGem CA and install it:

```
@@powershellconsole iwr -uri https://www.geotrust.com/resources/root\_certificates/certificates/GeoTrust\_Global\_CA.pem -OutFile c:\temp\GeoTrustCA.pem certutil -v -addstore Root C:\temp\GeoTrustCA.pem
```

2. In the `psft_customizations.yaml` file, add a new hash under each site hash called `text_properties:`. Create a new row with the key `138` and the value `"psadmin.io %{:app} Signon"`.

```

site_list:
  "%{hiera('pia_site_name')}":
    appserver_connections: "%{hiera('pia_psserver_list')}}"
    domain_conn_pwd:      "%{hiera('domain_conn_pwd')}}"
    webprofile_settings:
      profile_name:      "%{hiera('pia_webprofile_name')}}"
      profile_user:      "%{hiera('webserver_user')}}"
      profile_user_pwd:  "%{hiera('webserver_pwd')}}"
    report_repository_dir: "%{hiera('report_repository_dir')}}"
    text_properties:
      '138':              "psadmin.io %{: :app} Signon"

```

3. Next, we need to read the `text_properties:` hash in `modules\dpk_lab\manifests\io_web.pp` and start a loop.

```

class dpk_lab::io_web {
  $pia_domain_list = hiera('pia_domain_list')
  $pia_domain_list.each | $domain_name, $pia_domain_info | {
    $site_list = $pia_domain_info['site_list']
    $site_list.each | $site_name, $site_info | {
      $ps_cfg_home = $pia_domain_info['ps_cfg_home_dir']
      $portal_path = "${ps_cfg_home}/webserv/${domain_name}/applications/peoplesoft/PORTAL.war"
      file { "${domain_name}-${site_name}-pia-logo":
        ensure => present,
        path => "${portal_path}/${site_name}/images/Header.png",
        source => "puppet:///modules/dpk_lab/dpk-lab-logo-${: :app}.png",
      }
      $text_properties = $site_info['text_properties']
      $text_properties.each | $key, $value | {
        # ini processing will go here!
      }
    } # end-site
  } # end-pia
}

```

4. Use the `ini_setting` type to change the individual configuration value in `text.properties`.

```

class dpk_lab::io_web {
  $pia_domain_list = hiera('pia_domain_list')
  $pia_domain_list.each | $domain_name, $pia_domain_info | {
    $site_list = $pia_domain_info['site_list']
    $site_list.each | $site_name, $site_info | {
      $ps_cfg_home = $pia_domain_info['ps_cfg_home_dir']
      $portal_path = "${ps_cfg_home}/webserv/${domain_name}/applications/peoplesoft/PORTAL.war"
      file { "${domain_name}-${site_name}-pia-logo":
        ensure => present,
        path => "${portal_path}/${site_name}/images/Header.png",
        source => "puppet:///modules/dpk_lab/dpk-lab-logo-${: :app}.png",
      }
      $text_properties = $site_info['text_properties']
      $text_properties.each | $key, $value | {
        ini_setting { "${domain_name}-${site_name}-text-${key}":
          ensure => present,
          path => "${portal_path}/WEB-INF/psftdocs/${site_name}/text.properties",
          setting => $key,
          value => $value,
          key_val_separator => '=',
        }
      }
    }
  }
}

```

```
    section      => '',
  }
}
} # end-site
} # end-pia
}
```

5. Let's run the `io_web` module and validate our changes.

```
PS C:\psft\dpk\puppet> puppet apply -e "include ::dpk_lab::io_web" `
  --confdir= .
  --environment=env0
Warning: ...
...
Notice: Compiled catalog for ec2amaz-j9rg7hr.ec2.internal in environment production in 0.19 seconds
Notice: /Stage[main]/Dpk_lab::Io_web/Ini_setting[hcmwin-ps-text-138]/value: value changed '[redacted sensitive information]' to '[redacted sensitive information]'
Notice: Applied catalog in 3.98 seconds
```

6. Restart the web server to verify the Title change.

```
PS C:\psft\dpk\puppet> restart-service Psft*Pia*
```

You can use the `inifile` module for the following files: `configuration.properties`, `integrationGateway.properties`, `setEnv.cmd`, `psappsrv.cfg`, `psprcs.cfg`. Any file that uses the `key=value` format will work with the `inifile` module.

Install and use an io module

There are a number of community driven psadmin.io puppet modules. For this lab we will focus on `psadminio-io_portalwar`.

1. Clone the module from GitHub.

```
PS> git clone git@github.com:psadmin-io/psadminio-io_portalwar.git modules/io_portalwar
```

2. Open the `dpk_lab\manifests\io_web.pp` profile and update to contain `io_portalwar`.

```
class dpk_lab::io_web {
  contain ::io_portalwar
  ...
}
```

3. Remove the `$text_properties` code from `io_web.pp` we added before.

```
# Delete or comment out
```

```
#$text_properties = $site_info['text_properties']
#$text_properties.each | $key, $value | {
#   ini_setting { "${domain_name}-${site_name}-text-${key}":
#     ensure       => present,
#     path         => "${portal_path}/WEB-INF/psftdocs/${site_name}/text.properties",
#     setting      => $key,
#     value        => $value,
#     key_val_separator => '=',
#     section      => '',
#   }
# }
```

4. Remove the `text.properties` hash in `psft_customizations.yaml` we added before.

```
# Delete or comment out
# io_portalwar::text_properties:
#   '138': "psadmin.io %{:app} Signon"
```

5. Add a new `io_portalwar::text_properties` hash at the top of the `psft_customizations.yaml` file.

```
---
io_portalwar::text_properties:
  "%{hiera('pia_domain_name')}":
  "%{hiera('pia_site_name')}":
  '138': 'psadmin.conf 2019 Signon'
```

6. Run `puppet apply` with the new io module changes in place.

```
PS C:\psft\dpk\puppet> puppet apply -e "include ::dpk_lab::io_web" `
  --confdir=. `
  --environment=env0
```

7. Restart the web server to verify the Title changed to `psadmin.conf 2019 Signon`.

```
PS C:\psft\dpk\puppet> restart-service Psft*Pia*
```

Make an update to an io module

Let's make a small change to the `io_portalwar` documentation module.

1. Update the `README.md` file in `modules/io_portalwar`.

```
# io_portalwar

This was updated by me for conf 2019!
```

2. OPTIONAL: Make other changes to `io_portalwar`.

```
# Select an issue to work on or make your own change!
github_issues:
  14: 'Move `io_portalwar` hashes into delivered hashes'
  15: 'conf2019 lab - Update README'
  16: 'TODO'
```

Create a Pull Request for an io module

We will make a Pull Request in GitHub to contribute our change.

1. Create a new branch.

```
PS> cd modules/io_portalwar
PS> git checkout -b conf2019/psalab-$( $env:NODENAME.substring($env:NODENAME.length-3,3))
```

2. Check the status and review changed files.

```
PS> git status
```

3. Add the changes and commit.

```
PS> git add --all
PS> git commit -m "your message - Fixes #15"
PS> git status
```

4. Push your changes to GitHub.

```
PS> git push origin conf2019/psalab-$( $env:NODENAME.substring($env:NODENAME.length-3,3))
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 347 bytes | 347.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote:
remote: Create a pull request for 'conf2019/psalab-100' on GitHub by visiting:
remote:   https://github.com/psadmin-io/psadminio-io_portalwar/pull/new/conf2019/psalab-100
remote:
To github.com:psadmin-io/psadminio-io_portalwar.git
```

[new branch] conf2019/psalab-100 -> conf2019/psalab-100

5. Open the `psadminio-io_portalwar` in GitHub, and click on `branches` to review.

```
@yam1
https://github.com/psadmin-io/psadminio-io_portalwar/branches
user: psalab
pass: conf2019    TODO - preload password in firefox?
```

6. Under `Active Branches`, find your recently added branch.
7. Click on `New pull request`.
8. Change the branch base from `production` to `conf2019/dvlp`
`base: conf2019/dvlp <= compare: conf2019/psalab-[LABNUM]`
9. Edit the title, body, labels, etc.
10. Click `Create pull request`.
11. Let your instructor know you have completed the PR and wait for it to be reviewed and merged.
12. Once reviewed, refresh the PR page if open.
 1. Otherwise return to GitHub and open the `Pull requests` page.
 2. Click on the `Closed` button, find your now closed PR and review.
13. Find the review comments, commit log and branch deletion info.